(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 704 792 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.04.1996 Bulletin 1996/14

(51) Int Cl.[6]: **G06F 3/12**

(21) Application number: **95306919.2**

(22) Date of filing: **29.09.1995**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **29.09.1994 US 315274**

(71) Applicant: **XEROX CORPORATION**
Rochester New York 14644 (US)

(72) Inventors:
- **Ambalavanar, Samuel D.**
  Rochester NY 14625 (US)
- **Romano, Kenneth D.**
  Webster NY 14580 (US)
- **Sanford, Ronnie E.**
  Webster NY 14580 (US)
- **Frumusa, Anthony M.**
  Penfield NY 14526 (US)
- **Diaz, Orlando**
  Rochester NY 14620 (US)

(74) Representative: **Goode, Ian Roy et al**
Rank Xerox Ltd
Patent Department
Parkway
Marlow Buckinghamshire SL7 1YL (GB)

(54) Method of managing memory allocation in a printing system

(57) A method of managing memory allocation in a printing system including the steps of creating a plurality of blocks (306) in a memory (24) and designating each block with an identifier. In response to a request from a client, a first set of identifiers, corresponding with a first set of blocks, is placed into a database (304) by a resource manager (300). The client then accesses the database and, by reference to the first set of identifiers, begins filling up the first set of blocks with image data. As each block is filled, the client transmits an interrupt signal to a controller (44). After a predesignated one of the first set of blocks has been filled, the controller causes the resource manager to place a second set of identifiers in the database so that the client can access the second set of identifiers as soon as it has completed filling the first set of blocks.

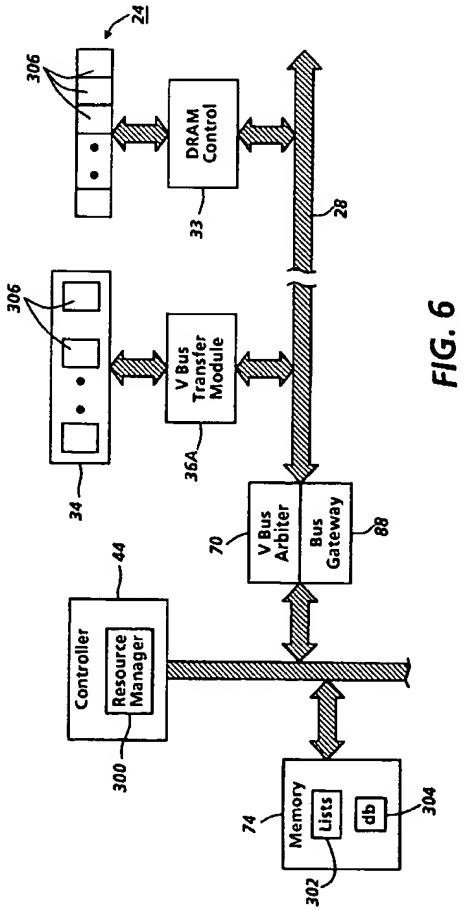

FIG. 6

EP 0 704 792 A1

Description

The present invention relates generally to a technique of memory management for a printing system and, more particularly, to a method of managing memory allocation for the printing system which minimizes both processing overhead and memory fragmentation.

Electronic printing systems typically include an input section, sometimes referred to as an input image terminal ("IIT"), a controller, sometimes referred to as an electronic subsystem ("ESS") and an output section or print engine, sometimes referred to as an image output terminal ("IOT"). In one type of electronic printing system, manufactured by Xerox® Corporation, known as the DocuTech® electronic printing system, a job can be inputted to the IIT from, among other sources, a network or a scanner. An example of an IIT with both network and scanner inputs is found in US-A-5,170,340.

Since digital printing systems store images electronically, a significant amount of memory is often required for storage. In a multifunctional digital printing system, various clients, i.e. various input/output devices of the printing system, seek to use the memory. That is, input clients seek to access the memory for storing image data and output clients seek to access the memory for the sake of consuming image data. Without some sort of arrangement for controlling employment of the memory by these clients, operation of the system can be impaired greatly. For example, a client with relatively slow processing capability can monopolize use of the memory at the expense of a client with relatively fast processing capability. Moreover, the memory needs of a group of clients may vary, among individual clients, over time. In the area of computer architecture, it is known that memory or resource management is a desirable approach for insuring that allocation of memory among a group of clients is performed in an orderly manner.

US-A-5,212,566 is directed toward a resource allocation scheme for a memory arrangement including disk and system memories, the system memory including a plurality of buffers. A system state controller communicates with the system memory, by way of a resource manager, and with a scanner, by way of a scan management arrangement, the scan management arrangement including a scan scheduler and a scan controller. In operation, the scanner, along with each client requesting use of the system memory, is allocated a set of buffers. During run time, the scanner fills buffers allocated to it with scanned data of a scan job, obtained by reading a document, and stores each filled buffer out to disk. If the scanner requires more buffers than are allocated to it, then a fault will occur. In response to the fault, a fault command flows from the scan controller to the system state controller, which system state controller, in turn, transmits a control command requesting the resource manager to adjust buffer allocation in the system memory. Under ideal circumstances, the scanner receives the buffers it needs to complete the scan job. As disclosed, reallocation includes obtaining a previously allocated buffer from a client other than the scanner.

While the resource management scheme of the '566 patent is well suited for its intended purpose, it appears to require a fair amount of processing overhead since buffer allocation among clients must often be assessed in obtaining a free buffer for the scanner. It would be desirable to provide a resource management scheme which uses a minimum amount of processing overhead in obtaining and providing memory for a given client. Additionally, in the resource management scheme of the '566, all of the buffers allocated to a given client may not be used fully. This can lead to underutilization of memory space and even memory fragmentation. It would be desirable to provide a resource management scheme in which all memory space of the system memory is employed in the most efficient manner possible.

It is an object of the present invention to provide such a scheme.

In accordance with one aspect of the present invention there is provided a method of managing memory allocation in a printing system with a controller and memory, the controller having a resource manager for managing use of the memory and the printing system supporting input clients, each input client seeking to store one or more images, in the form of image data, in the memory, comprising the steps of: partitioning at least a portion of the memory to create a plurality of blocks; providing each of the plurality of blocks with an identifier, each identifier indicating a location of a block in the memory; in response to a request from a selected input client, placing a first set of identifiers, corresponding with a first set of blocks, in a database; accessing the first set of identifiers with the selected input client; filling up one or more of the first set of blocks, with image data, by referring to the first set of identifiers; transmitting an interrupt signal to the controller, with the selected input client, each time one of the first set of blocks is filled; and in response to a designated one of the blocks being filled, placing a second set of identifiers, corresponding with a second set of blocks, in the database, so that when the selected input client has filled up a last one of the first set of blocks, the selected input client accesses the second set of blocks, by reference to the second set of identifiers and begins filling up a first block of the second set of blocks wherein memory allocation is accomplished with a minimum amount of communication between various components of the printing system.

In accordance with another aspect of the present invention there is provided a method of managing memory allocation in a printing system with an input client, the input client storing image data in memory for outputting an image associated with the image data, comprising the steps of: providing to the input client, a first set of blocks including a first block and a second block, wherein the second block includes a first part and a second part; filling up both the first block and the first part of the second block with image data, an end of the first part of the sec-

ond block representing a corresponding end of a first image; and filling up the second part of the second block with image data, the image data in the second part of the second block corresponding with a second image, wherein the first image is different than the second image and usage of memory space for storing image data is maximized.

These and other aspects of the invention will become apparent from the following description, the description being used to illustrate a preferred embodiment of the invention when read in conjunction with the accompanying drawings, in which:-

Figure 1 is a block diagram depicting a multifunctional, network adaptive printing machine;
Figure 2 is a block diagram of a video control module for the printing machine of Figure 1;
Figure 3 is a block diagram of a transfer module used in conjunction with the printing machine of Figure 2;
Figure 4 is a block diagram of a facsimile card used in conjunction with the printing machine of Figure 2;
Figure 5 is a block diagram of a network controller for the printing machine of Figure 1;
Figure 6 is a block diagram of a resource management scheme including a selected number of components from the block diagram of Figure 2;
Figures 7-10 is a flow diagram illustrating some of the functionality of the resource management scheme of Figure 6;
Figure 11 is a schematic representation of electronic precollation (EPC) memory being used in conjunction with the resource management scheme of the present invention;
Figure 12 is a schematic representation illustrating how partial blocks are employed, in the resource management scheme, to reduce memory fragmentation;
Figure 13 is a schematic representation of a database format used in conjunction with the resource management scheme; and
Figures 14-17 are schematic representations illustrating how a combination of electronic precollation (EPC) and disk memory is used in conjunction with the resource management scheme.

Referring to Figure 1, a multifunctional, network adaptive printing system is designated by the numeral 10. The printing system 10 includes a printing machine 12 operatively coupled with a network service module 14. The printing machine 12 includes an electronic subsystem 16, referred to as a video control module (VCM), communicating with a scanner 18 and a printer 20. In one example, the VCM 16, which will be described in further detail below, coordinates the operation of the scanner and printer in a digital copying arrangement. In a digital copying arrangement, the scanner 18 (also referred to as image input terminal (IIT)) reads an image on an original document by using a CCD full width array and converts analog video signals, as gathered, into digital signals. In turn, an image processing system 22 (Figure 2), associated with the scanner 18, executes signal correction and the like, converts the corrected signals into multi-level signals (e.g. binary signals), compresses the multi-level signals and preferably stores the same in electronic precollation (EPC) memory 24.

Referring again to Figure 1, the printer 20 (also referred to as image output terminal (IOT)) preferably includes a xerographic print engine. In one example, the print engine has a multi-pitch belt (not shown) which is written on with an imaging source, such as a synchronous source (e.g. laser raster output scanning device) or an asynchronous source (e.g. LED print bar). In a printing context, the multi-level image data is read out of the EPC memory 24 (Figure 2) while the imaging source is turned on and off, in accordance with the image data, forming a latent image on the photoreceptor. In turn, the latent image is developed with, for example, a hybrid jumping development technique and transferred to a print media sheet. Upon fusing the resulting print, it may be inverted for duplexing or simply outputted. It will be appreciated by those skilled in the art that the printer can assume other forms besides a xerographic print engine without altering the concept upon which the disclosed embodiment is based. For example, the printing system 10 could be implemented with a thermal ink jet or ionographic printer.

Referring specifically to Figure 2, the VCM 16 is discussed in further detail. The VCM 16 includes a video bus (VBus) 28 with which various I/O, data transfer and storage components communicate. Preferably, the VBus is a high speed, 32 bit data burst transfer bus which is expandable to 64 bit. The 32 bit implementation has a sustainable maximum bandwidth of approximately 60 MBytes/sec. In one example, the bandwidth of the VBus is as high as 100 MBytes/sec.

The storage components of the VCM reside in the EPC memory section 30 and the mass memory section 32. The EPC memory section includes the EPC memory 24, the EPC memory being coupled with the VBus by way of a DRAM controller 33. The EPC memory, which is preferably DRAM, provides expansion of up to 64 MBytes, by way of two high density 32 bit SIMM modules. The mass memory section 32 includes a SCSI hard drive device 34 coupled to the VBus by way of a transfer module 36a. As will appear, other I/O and processing components are coupled respectively to the VBus by way of transfer modules 36. It will be appreciated that other devices (e.g. a workstation) could be coupled to the VBus by way of the transfer module 36a through use of a suitable interface and a SCSI line.

Referring to Figure 3, the structure of one of the transfer modules 36 is discussed in further detail. The illustrated transfer module of Figure 3 includes a packet buffer 38, a VBus interface 40 and DMA transfer unit 42 . The transfer module 36, which was designed with "VHSIC" Hardware Description Language (VHDL), is a pro-

grammable arrangement permitting packets of image data to be transmitted along the VBus at a relatively high transfer rate. In particular, the packet buffer is programmable so that the segment or packet can be varied according to the available bandwidth of the VBus. In one example, the packet buffer can programmed to handle packets of up to 64 Bytes Preferably, the packet size would be reduced for times when the VBus is relatively busy and increased for times when activity on the bus is relatively low.

Adjustment of the packet size is achieved with the VBus interface 40 and a system controller 44 (Figure 5). Essentially, the VBus interface is an arrangement of logical components, including, among others, address counters, decoders and state machines, which provides the transfer module with a selected degree of intelligence. The interface 40 communicates with the system controller to keep track of desired packet size and, in turn, this knowledge is used to adjust the packet size of the packet buffer 38, in accordance with bus conditions. That is, the controller, in view of its knowledge regarding conditions on the VBus 28, passes directives to the interface 40 so that the interface can adjust packet size accordingly. Further discussion regarding operation of the transfer module 36 is provided below

More particularly, each DMA transfer unit employs a conventional DMA transfer strategy to transfer the packets. In other words, the beginning and end addresses of the packet are used by the transfer unit in implementing a given transfer. When a transfer is complete, the interface 40 transmits a signal back to the system controller 44 so that further information, such as desired packet size and address designations, can be obtained.

Referring to Figures 1 and 2, three I/O components are shown as being coupled operatively to the VBus 28, namely a FAX module 48, the scanner or IIT 18, and the printer or IOT 20; however, it should be recognized that a wide variety of components could be coupled to the VBus by way an expansion slot 50. Referring to Figure 4, an implementation for the FAX module, which is coupled to the VBus 28 by way of transfer module 36b, is discussed in further detail. In the preferred embodiment, a facsimile device (FAX) 51 includes a chain of components, namely a section 52 for performing Xerox adaptive compression/decompression, a section 54 for scaling compressed image data, a section 56 for converting compressed image data to or from CCITT format, and a modem 58, preferably manufactured by Rockwell Corporation, for transmitting CCITT formatted data from or to a telephone, by way of a conventional communication line.

Referring still to Figure 4, each of the sections 52, 54 and 56 as well as modem 58 are coupled with the transfer module 36b by way of a control line 60. This permits transfers to be made to and from the FAX module 48 without involving a processor. As should be understood, the transfer module 36b can serve as a master or slave for the FAX module in that the transfer module can provide image data to the FAX for purposes of transmission or receive an incoming FAX. In operation, the transfer module 36b reacts to the FAX module in the same manner that it would react to any other I/O component. For example, to transmit a FAX job, the transfer module 36b feeds packets to the section 52 through use of the DMA transfer unit 42 and, once a packet is fed, the transfer module transmits an interrupt signal to the system processor 44 requesting another packet. In one embodiment, two packets are maintained in the packet buffer 38 so that "ping-ponging" can occur between the two packets. In this way, the transfer module 36b does not run out of image data even when the controller cannot get back to it immediately upon receiving an interrupt signal.

Referring again to Figure 2, the IIT 18 and IOT 20 are operatively coupled to the VBus 28 by way of transfer modules 36c and 36d. Additionally, the IIT 18 and the IOT 20 are operatively coupled with a compressor 62 and a decompressor 64, respectively. The compressor and decompressor are preferably provided by way of a single module that employs Xerox adaptive compression devices. Xerox adaptive compression devices have been used for compression/decompression operations by Xerox Corporation in its DocuTech® printing system. In practice, at least some of the functionality of the transfer modules is provided by way of a 3 channel DVMA device, which device provides local arbitration for the compression/decompression module.

As further illustrated by Figure 2, the scanner 18, which includes the image processing section 22, is coupled with an annotate/merge module 66. Preferably the image processing section includes one or more dedicated processors programmed to perform various desired functions, such as image enhancement, thresholding/screening, rotation, resolution conversion and TRC adjustment. The selective activation of each of these functions can be coordinated by a group of image processing control registers, the registers being programmed by the system controller 44. Preferably, the functions are arranged along a "pipeline" in which image data is inputted to one end of the pipe, and image processed image data is outputted at the other end of the pipe. To facilitate throughput, transfer module 36e is positioned at one end of the image processing section 22 and transfer module 36c is positioned at another end of the section 22. As will appear, positioning of transfer modules 36c and 36e in this manner greatly facilitates the concurrency of a loopback process.

Referring still to Figure 2, arbitration of the various bus masters of the VCM 16 is implemented by way of a VBus arbiter 70 disposed in a VBus arbiter/bus gateway 71. The arbiter determines which bus master (e.g. FAX module, Scanner, Printer, SCSI Hard Drive, EPC Memory or Network Service Component) can access the VBus at one given time. The arbiter is made up of two main sections and a third control section. The first section, i.e., the "Hi-Pass" section, receives input bus re-

quests and current priority selection, and outputs a grant corresponding to the highest priority request pending. The current priority selection input is the output from the second section of the arbiter and is referred to as "Priority Select". This section implements priority rotation and selection algorithm. At any given moment, the output of the logic for priority select determines the order in which pending requests will be serviced. The input to Priority Select is a register which holds an initial placement of devices on a priority chain. On servicing requests, this logic moves the devices up and down the priority chain thereby selecting the position of a device's next request. Control logic synchronizes the tasks of the Hi-Pass and the Priority Select by monitoring signals regarding request/grant activity. It also prevents the possibility of race conditions.

Referring to Figure 5, the network service module 14 is discussed in further detail. As will be recognized by those skilled in the art, the architecture of the network service module is similar to that of a known "PC clone". More particularly, in the preferred embodiment, the controller 44, which preferably assumes the form of a SPARC processor, manufactured by Sun Microsystems, Inc., is coupled with a standard SBus 72. In the illustrated embodiment of Figure 5, a host memory 74, which preferably assumes the form of DRAM, and a SCSI disk drive device 76 are coupled operatively to the SBus 72. While not shown in Figure 5, a storage or I/O device could be coupled with the SBus with a suitable interface chip. As further shown in Figure 5, the SBus is coupled with a network 78 by way of an appropriate network interface 80. In one example, the network interface includes all of the hardware and software necessary to relate the hardware/software components of the controller 44 with the hardware/software components of the network 78. For instance, to interface various protocols between the network service module 14 and the network 78, the network interface could be provided with, among other software, Netware® from Novell Corp.

In one example, the network 78 includes a client, such as a workstation 82 with an emitter or driver 84. In operation, a user may generate a job including a plurality of electronic pages and a set of processing instructions. In turn, the job is converted, with the emitter, into a representation written in a page description language, such as PostScript. The job is then transmitted to the controller 44 where it is interpreted with a decomposer, such as one provided by Adobe Corporation.

Referring again to Figure 2, the network service module 14 is coupled with the VCM 16 via a bus gateway 88 of the VBus arbiter/bus gateway 71. In one example, the bus gateway comprises a field programmable gate array provided by XILINX corporation. The bus gateway device provides the interface between the host SBus and the VCM VBus. It provides VBus address translation for accesses to address spaces in the VBus real address range, and passes a virtual address to the host SBus for virtual addresses in the host address range. A DMA channel for memory to memory transfers is also implemented in the bus gateway. Among other things, the bus gateway provides seamless access between the VBus and SBus, and decodes virtual addresses from bus masters, such as one of the transfer modules 36, so that an identifier can be obtained from a corresponding slave component. It will be appreciated by those skilled in the art that many components of the printing system 10 are implemented in the form of a single ASIC.

Referring to Figures 2, 3 and 5, further discussion regarding DMA transfer of each of the transfer modules 36 is provided. In particular, in one example, the images of a job are stored in the host memory 74 as a series of blocks. Referring to Figure 16, a series of blocks is shown as being stored in the EPC memory 24. Preferably, each block comprises a plurality of packets. In operation, one of the transfer modules 36 is provided, by the controller 44, with the beginning address of a block and the size of the block. In turn, for that block, the transfer module 36 effects a packet tranfer and increments/decrements a counter. This procedure is repeated for each packet of the block until the interface 40 determines, by reference to the counter, that the last packet of the block has been transferred. Typically, for each stored image, several blocks are transferred, in a packet-by-packet manner, as described immediately above.

Referring to Figure 6 a scheme for managing memory allocation, i.e. a resource management scheme, is illustrated. More particularly, the controller 44 includes a resource manager 300, while the host memory 74 includes a pair of lists 302, referred to respectively as the "free block list" and the "free partial block list", and a database ("db") 304. The resource manager is implemented by way of suitable algorithms, the details of which will be discussed in further detail below, and the significance of the lists and the database, relative to the resource management scheme, will also be discussed below. Additionally, the EPC memory 24 and the SCSI hard drive ("disk") 34 are shown as being comprised of blocks 306. A discussion of a methodology for forming and allocating memory blocks follows:

Referring to Figures 6-10, the algorithms for implementing the resource management scheme are discussed. Initially, at step 308, the EPC memory 24 is partitioned into a series of the blocks 306. A partitioned set of memory blocks is also shown in Figure 11. Preferably, the block size is varied in accordance with factors, such as image size to be stored. For example, if a location generally copies complex documents which results in poorly compressed (large) image, the block size can be increased. As will appear from the discussion below, increased block size will result in fewer interrupts by a client (e.g. scanner 18) of the controller 44.

Each block is then provided with identification information (step 310), such as block ID, block address and block size, which identification information is placed, at step 312, in the free block list. Preferably, each list in the host memory 74 is a linked list of structures. At step 316,

the resource management system waits for a memory request from a client. In the present context, a client is an input or output device encompassed by the printing system 10 (Figures 1,2 and 5). A client initiates a request by transmitting a suitable request or interrupt signal to the controller 44. Upon receiving a request signal, the controller determines, via step 316, whether the client is an input client. If the client is an input client, then the process proceeds to step 318, otherwise the process proceeds directly to step 320 (Figure 9) where an output client request is serviced.

Assuming the requesting client is an input client, the resource manager 300 examines the free partial block list to determine if a partial block is available for the requesting client. Referring to Figure 12, an example of the allocation of a partial block to the beginning of an image will be discussed. In particular, at system initialization, no partial block is available for an image 1 of a job. After image data for image 1 is delivered to the memory, however, a partial, unfilled block may remain. As shown in Figure 12, and explained in further detail below, the partially unfilled block, with its corresponding identifier is made available for use with the next image.

Returning to Figure 7, if a partial, unfilled block is available, then it is designated with an identifier and, at step 322, placed in the db 304. Next, at step 324, the resource manager consults the free block list to determine if a nominal number of blocks are available for use by the input client. In the preferred embodiment, each client is assigned a value corresponding to the number of nominal blocks to which it is entitled. In one example, assignment is based on the processing speed of the requesting client. That is, per each request, it may be desirable to provide fast processing clients with more blocks than slow processing clients. In one situation, the nominal number of blocks to be assigned a requesting client may not be available in the free block list. In this situation, the resource manager may provide the requesting client with one or more partial blocks until a whole block becomes available.

Assuming the nominal number of blocks is available, at step 326 the resource manager will place appropriate identifiers (i.e. information identifying both a first address and a size of each block) in the db 304. Referring to Figure 13, a suitable database structure for use with the disclosed embodiment is shown. The database is constructed in a hierarchical scheme in which jobs are linked to images and images are linked to blocks. In one example, where the client's storable image data is associated with a first image (i.e. "Image 1) of a first job (i.e. "JOB 1"), then the first block identifier is placed at the location designated as "Block 1 Address". Subsequently, the client will access the database and, at step 328 (Figure 7), locate the address of the first available block. The client will then, in cooperation with, for example, one of transfer modules 36, fill up the located block. When the scanner is serving as the client, the scanner will initiate a DMA transfer, with EPC memory 24, via the transfer module 36D (Figure 2). Referring again to Figure 14, the scanner is shown as using the EPC memory in conjunction with other clients. While the block 306A is shown as being a whole block, it will be understood that, in many instances, it would be a partial block.

The printing system 10 offers the advantageous feature of storing jobs, intended to be outputted as multiple sets, on disk. In this way EPC memory can be made available to multiple clients in a relatively short time interval. Referring to step 332 of Figure 8, when disk storage is desired, each stored block is copied to disk 34 (also see step 334 of Figure 8). Referring to Figure 14, a graphic representation demonstrating the relationship between EPC memory and disk is provided.

As further shown in Figure 14, preferably, a minimum amount of input image pages, intended for printing, are buffered prior to printing. This has been found to be advantageous since a printer typically processes image data at a rate much greater than that of most input clients, such as the scanner. In the illustrated embodiment of Figure 15, a variable buffer zone 336 is maintained for the scan client. This buffer zone is used to move image directly to disk, which enables the system to continue scanning without stopping. It will be appreciated that the variable zone can be used by clients, other than the scanner 18, to facilitate storage.

Referring again to Figure 8, the input client transmits an interrupt signal to the controller, at step 338, when a block has been filled with image data. Alternatively, the input client could be provided, in advance, with pointers to lists of block addresses. In this way, the input client would read, without controller intervention, the locations of blocks to be used.

A determination is made at step 340 as to whether a full image has been written into EPC memory 24. Assuming that the end of the image has not been reached, it is determined, at step 342, whether another nominal number of blocks is required. It should be appreciated that, typically, when a client requires a nominal number of blocks, the resource manager provides it with a set of plural blocks. In application, those blocks follow a sequence and one of the blocks in the sequence is identified as a "relative last block" which, when reached, indicates that another set of blocks is required. The position of the relative last block is variable in that it need not, in absolute terms, be the last block of the set. If the relative last block has not been reached, then the process loops back to step 328 where the db 304 (Figure 6) is accessed so that the client can locate the next block to be filled. On the other hand, if the relative last block has been reached, then the process loops back to step 324 for obtaining at least a part of another block set.

Referring still to Figure 8, if it is determined, at step 340, that a full image has just been written into memory, then a series of steps is performed to prepare for the receiving of another image. First, the resource manager 300 (Figure 6) determines, with step 344, if all of the full blocks have been used by the input client. If not, then the

identifier of each surplus whole block is placed in the free block list (step 348), otherwise the process proceeds to step 350 where the resource manager determines if the image ends on a partial block. Referring again to Figure 12, an example of how an image might end at a partial block is shown for the "Image 1". In the preferred embodiment, the size of the unused part of Image 1 is then determined in accordance with step 352 of Figure 8. Referring to Figure 9, if the size of the partial block is greater than a selected minimum size (step 354), then an identifier is assigned to the partial block (step 356) and placed in the free partial block list so that the partial block can be used to receive image data from another image, such as the "Next Image" of Figure 12. For those cases in which a given partial block is smaller than a selected minimum, the given partial block is saved for "garbage collection", the significance of which will be described below.

At steps 320 and 360, the preferred methodology accommodates for the needs of an output client, such as a printer. Regarding step 360, the output client is preferably "told" where the image data, intended for use in outputting, resides. In this way, the output client can read the image data from the EPC memory. Additionally, as shown in Figure 14, an output operation can be executed just before or after an input operation.

Referring to Figures 14, 16 and 17, an application of the the present memory management scheme, with respect to the printing client, is discussed in further detail. In the illustrated embodiments of Figures 14, 16 and 17, a given job, intended to be printed in multiple sets, is shown as including six images. In Figure 14, the first three images are buffered and copied to disk. In Figure 16, writing of images, to memory, continues concurrent with the reading of first and second image blocks by the printer. While the read/write operations are not "concurrent", in absolute terms, they appear, to a system operator, as being concurrent.

In Figure 17, the end of the job is written into EPC memory at blocks 306B, 306C and 306D, while the beginning of the printing of a second set is initiated at block 306E. For the printing of the second set, the image 2, along with the block for 1D need not be copied from disk. As should be recognized, the EPC memory and disk function in a manner comparable to a ring buffering arrangement in that image data from disk can be written over image data in the EPC memory, continually, in order to form a desired number of sets.

At step 362 (Figure 9), it is determined whether an appropriate time has arrived for "garbage collection". In the present context, garbage collection refers to combining "spent blocks", i.e. blocks having image data already "consumed" by an output client, for future use. In one example, a check for garbage collection is performed after a predesignated number of images have been printed. More particularly, garbage collection is performed as a background task, i.e. during a noncritical time of a job cycle.

To implement garbage collection, the possibility of block combination is checked at step 364 and partial blocks are combined, if possible, at step 366. It follows that block combination constitutes, in one example, linking partial blocks with references. As blocks are formed from partial blocks (step 368), some partial block identifiers will be discarded and the resulting whole block will be placed in the free block list. If garbage collection is not performable, the process proceeds to step 372.

At step 372, a check is performed to determine if the currently completed image is the last image in the job. If the image is not the last image, then the process loops back to step 316 where the input/output client accesses the db 304 for another block identifier, assuming that the client is ready. If, on the other hand, the job is complete, then a determination is made, at step 374, as to whether repartitioning is required. Repartitioning is performed (step 376) until a suitable block size is obtained. Subsequent to repartitioning the process loops back to step 316.

Numerous features of the above-disclosed embodiment will be appreciated by those skilled in the art:

First, the disclosed resource management scheme functions with a minimum amount of processing overhead. In particular, per a request by a client, block identifiers are placed at selected locations in a database. In turn, a selected client can access the database, determine where available blocks exist in memory, by reference to the block identifiers, and begin filling those blocks. As the client fills the blocks, it signals a controller and, in response to the signals, the controller indicates to a resource manager when to place more block identifiers in the database for the selected client. The process of providing blocks is transparent to the selected client and the resource manager is not required to possesses any significant knowledge about memory allocations of system clients in order to service the selected client appropriately.

Second, memory utilization of the system is enhanced in that memory blocks are employed in a particularly efficient manner. That is, unused blocks, whether they be partial or whole blocks, are placed in one or more free lists as soon as a given client ceases to have an immediate need for them. In turn, unused free blocks are allocated to other clients, in an expeditious manner and partial blocks are, in many instances, used to store the beginning and end parts of an image. In this way memory fragmentation is minimized and memory space is made available to clients, who have an urgent demand for it, as soon as possible.

Third, the resource management technique is flexible in that a wide variety of system parameters, such as nominal block size, number of blocks allocated to a given client at one time, and block allocation timing are variable. In this way, the attendant printing system can accommodate for varying input/output demands. Preferably, the system can keep track of compression ratios and adjust the variables to maximize performance for a par-

ticular location's majority usage. For example, if a location generally copies complex documents, which results in poorly compressed (large) images, the block size can be increased. Use of increased block size will result in fewer interrupts to the controller. Moreover, the system can, among other things, adjust block allocation according to individual client processing capability and predict the moment at which blocks, for a given client, should be made available.

Finally, the resource management scheme can be used to coincidentally manage volatile and nonvolatile memory in a manner that maximizes the functionality of the volatile memory, which volatile memory may be limited in space. Through efficient management of volatile and nonvolatile memory, even a job, having a size greater than that of the volatile memory (e.g. EPC memory) can be outputted in multiple sets with relatively little degradation in output rate. Moreover, volatile and nonvolatile memory can be used, conjunctively, to insure that even complex documents, having much greater size than that of the volatile memory, can be stored by a given input client without impairing operation of that given input client.

Claims

1. A method of managing memory allocation in a printing system with a controller and memory, the controller having a resource manager for managing use of the memory and the printing system supporting input clients, each input client seeking to store one or more images, in the form of image data, in the memory, comprising the steps of:

   partitioning at least a portion of the memory to create a plurality of blocks;
   providing each of the plurality of blocks with an identifier, each identifier indicating a location of a block in the memory;
   in response to a request from a selected input client, placing a first set of identifiers, corresponding with a first set of blocks, in a database;
   accessing the first set of identifiers with the selected input client;
   filling up one or more of the first set of blocks, with image data, by referring to the first set of identifiers;
   transmitting an interrupt signal to the controller, with the selected input client, each time one of the first set of blocks is filled; and
   in response to a designated one of the blocks being filled, placing a second set of identifiers, corresponding with a second set of blocks, in the database, so that when the selected input client has filled up a last one of the first set of blocks, the selected input client accesses the second set of blocks, by reference to the second set of identifiers and begins filling up a first block of the second set of blocks wherein memory allocation is accomplished with a minimum amount of communication between various components of the printing system.

2. The method of claim 1, wherein said filling step includes filling the first set of blocks with image data generated from a scan service, the scan service including a scanning apparatus for converting information disposed on a document, to the image data.

3. The method of claim 1, wherein the first set of blocks includes a last block with said step of placing second set of identifiers including placing the second set of identifiers into the database after the last block has been filled with image data.

4. The method of claim 1, further comprising the step of outputting a corresponding job, including one or both of the first and second block sets, with an output client.

5. The method of claim 1, in which all of a filled block is used to print part of a corresponding job, further comprising the step of placing an identifier, associated with the used, filled block, in a first list.

6. The method of claim 1 further comprising the step of repartitioning the memory, after a selected number of blocks have been filled, for changing block size in accordance with varying image size of individual images being stored in memory.

7. A method of managing memory allocation in a printing system with an input client, the input client storing image data in memory for outputting an image associated with the image data, comprising the steps of:

   providing to the input client, a first set of blocks including a first block and a second block, wherein the second block includes a first part and a second part;
   filling up both the first block and the first part of the second block with image data, an end of the first part of the second block representing a corresponding end of a first image; and
   filling up the second part of the second block with image data, the image data in the second part of the second block corresponding with a second image, wherein the first image is different than the second image and usage of memory space for storing image data is maximized.

8. The method of claim 7, further comprising the steps of:

designating the second part of the second block with an identifier;
placing the identifier in a partial block list, said second part filling up step including accessing the partial block list, with the input client, to determine a space, in memory, to which the image data of the second image is to be transmitted.

9. The method of claim 7, in which the first set of blocks includes a third block and the third block remains unfilled subsequent to the second part being filled further comprising the steps of:

designating the third block with an identifier; and accessing the third block identifier with the input client for filling the third block with image data from the second image.

10. The method of claim 7, in which the input client possesses a processing capability and the first set of blocks includes a selected number of blocks, further comprising the step of adjusting the selected number of blocks as a function of the input client processing capability.

11. The method of claim 7, wherein said filling step includes filling the first block and the first part of the second block with image data generated from a scan service, the scan service including a scanning apparatus for converting information disposed on a document, to the image data.

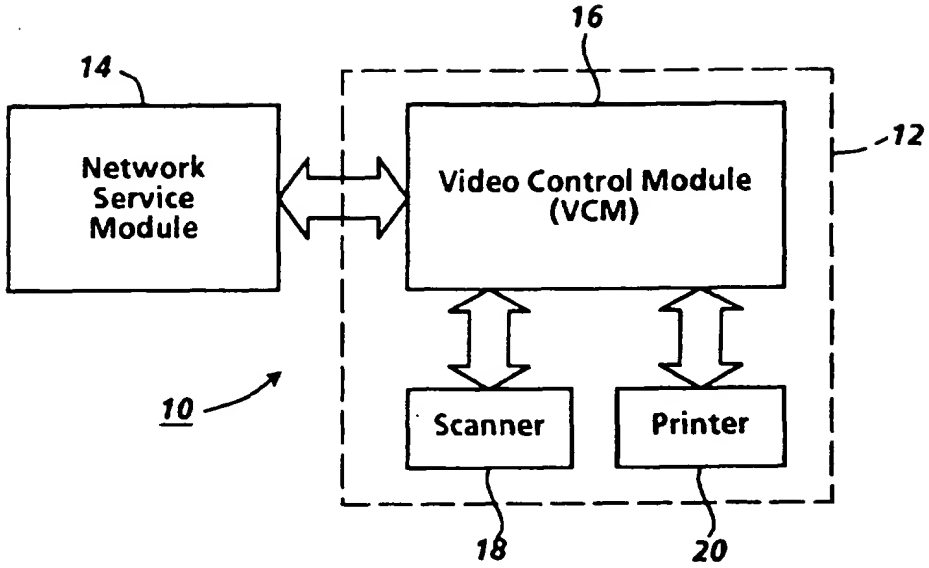
16
14
12
Network Service Module
Video Control Module (VCM)
10
Scanner
Printer
18
20

FIG. 1

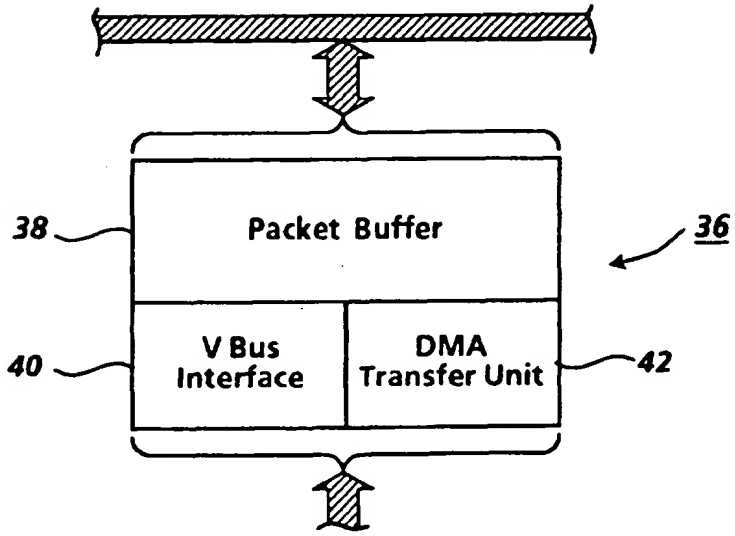
38
Packet Buffer
36
40
V Bus Interface
DMA Transfer Unit
42

FIG. 3

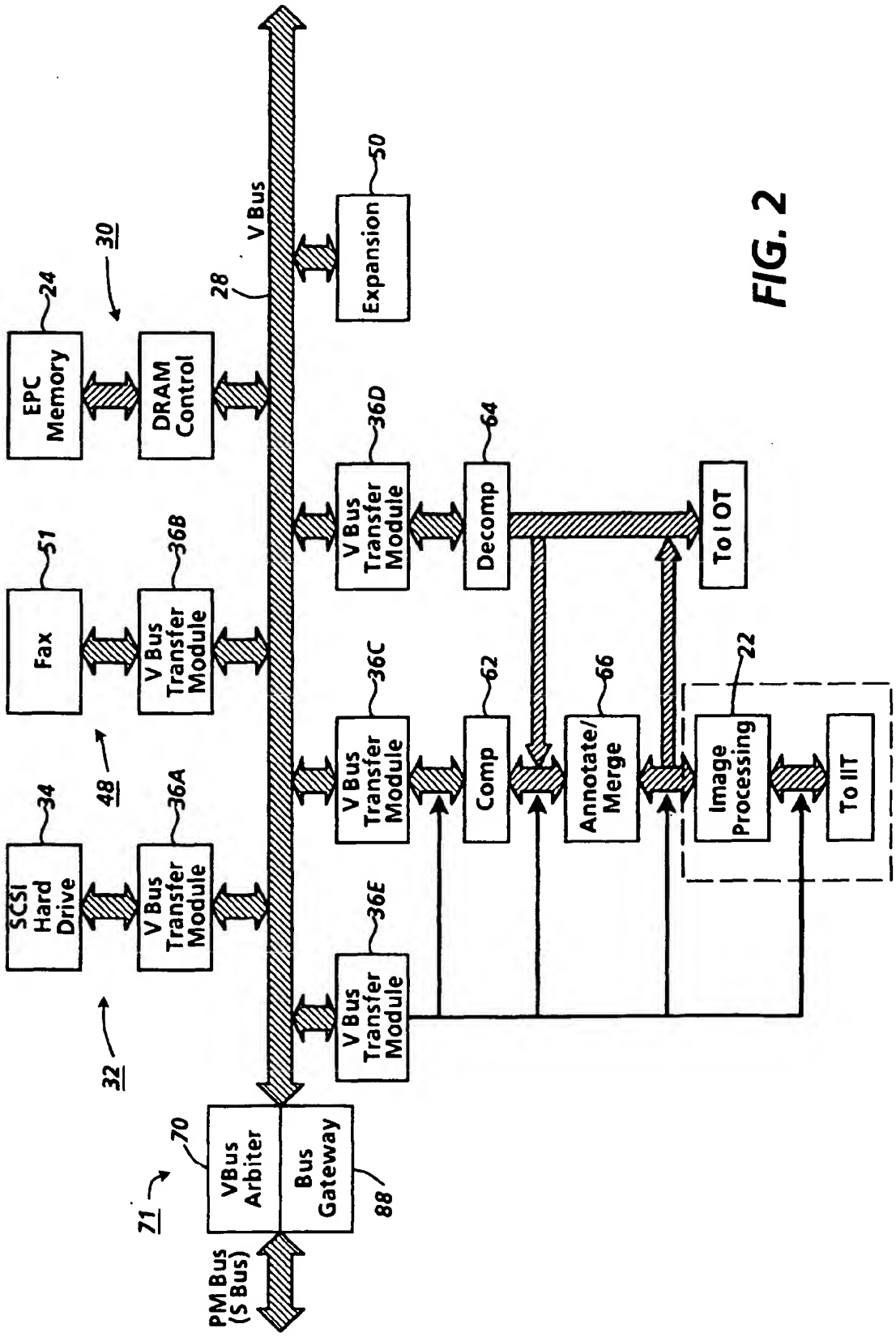
PM Bus (S Bus)
71
70
VBus Arbiter
Bus Gateway
88
32
SCSI Hard Drive
34
V Bus Transfer Module
36A
48
Fax
51
V Bus Transfer Module
36B
EPC Memory
24
DRAM Control
30
28
V Bus
Expansion
50
V Bus Transfer Module
36D
Decomp
64
To IOT
V Bus Transfer Module
36C
Comp
62
Annotate/ Merge
66
22
Image Processing
To IIT
V Bus Transfer Module
36E

FIG. 2

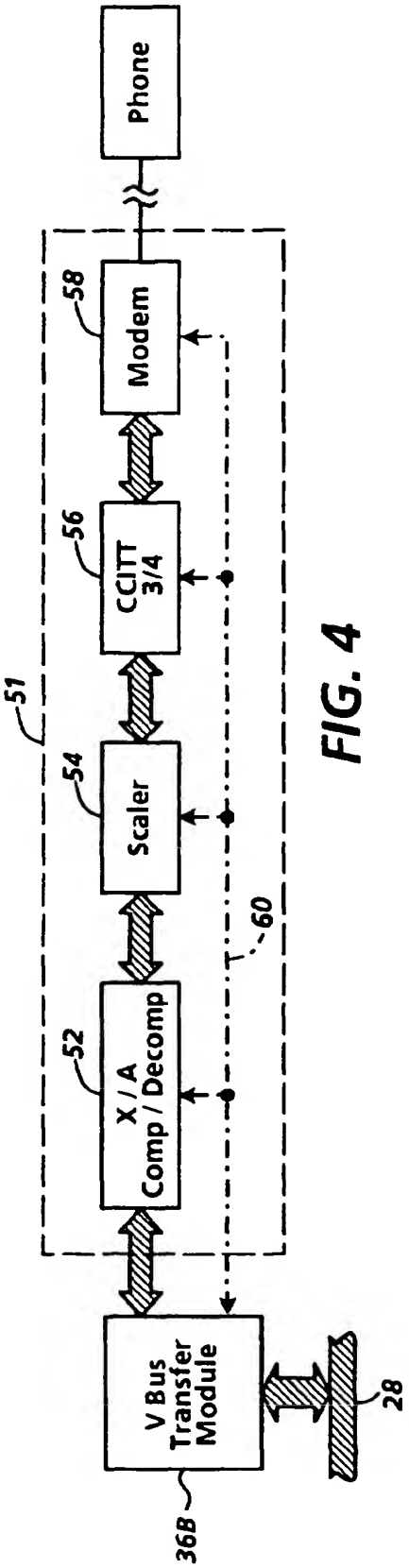
Phone
58
Modem
56
CCITT 3/4
54
Scaler
60
52
X / A Comp / Decomp
51
V Bus Transfer Module
36B
28

FIG. 4

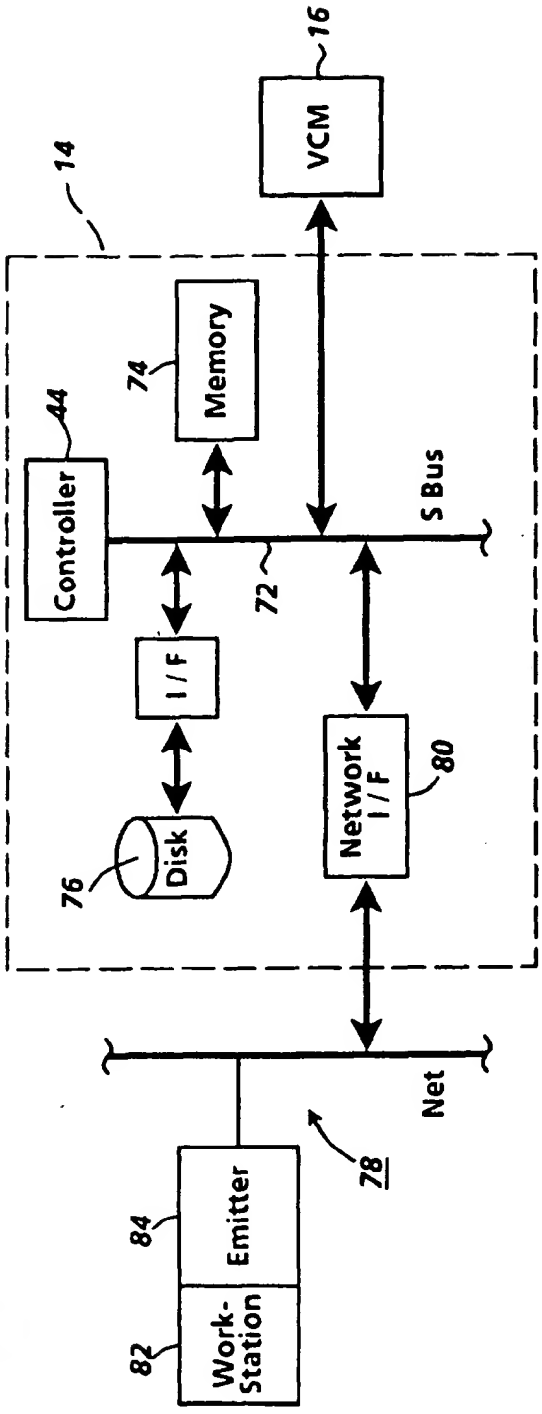
16
VCM
14
74
Memory
44
Controller
S Bus
72
I / F
76
Disk
Network I / F
80
Net
78
84
Emitter
82
Work-Station

FIG. 5

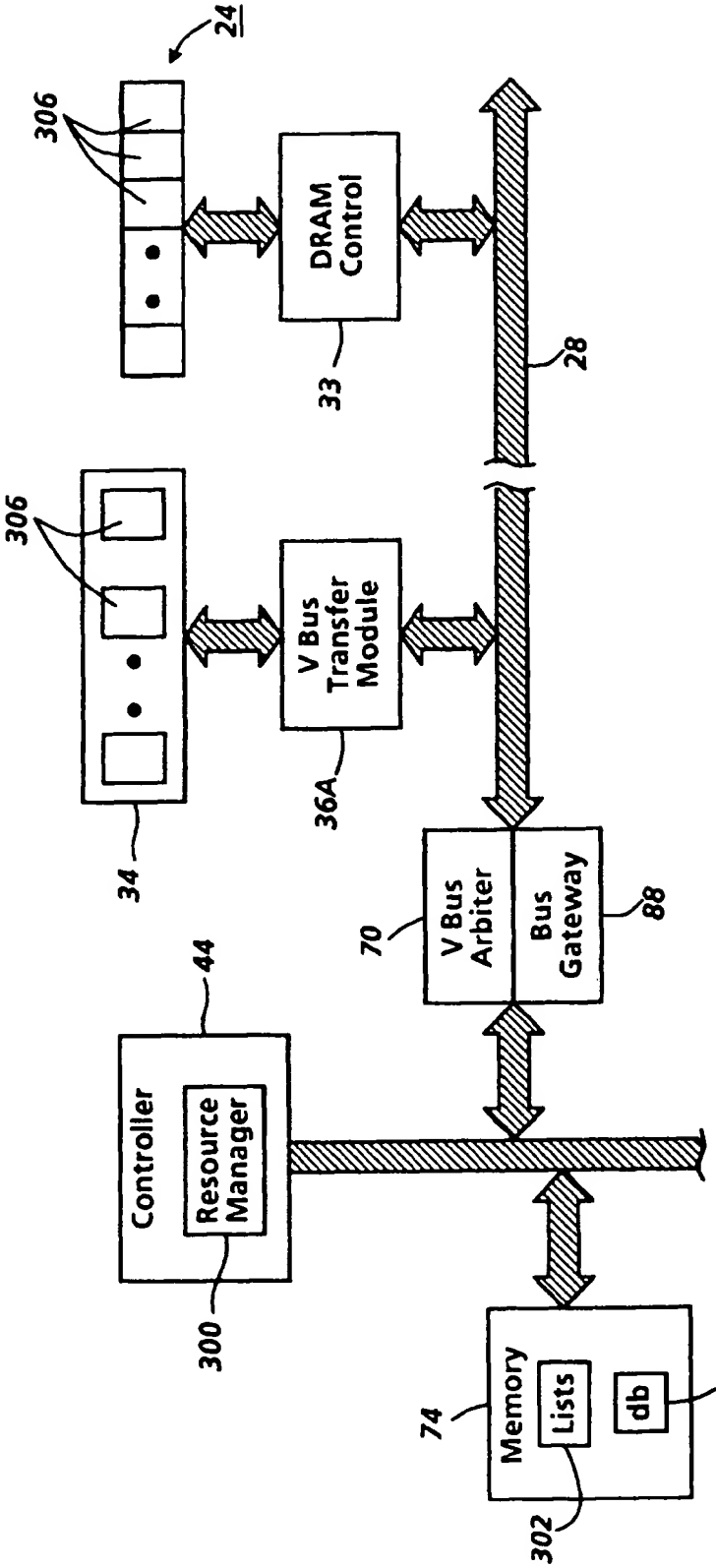
24
306
DRAM Control
33
28
306
V Bus Transfer Module
36A
34
70
V Bus Arbiter
Bus Gateway
88
44
Controller
Resource Manager
300
74
Memory
Lists
db
304
302

FIG. 6

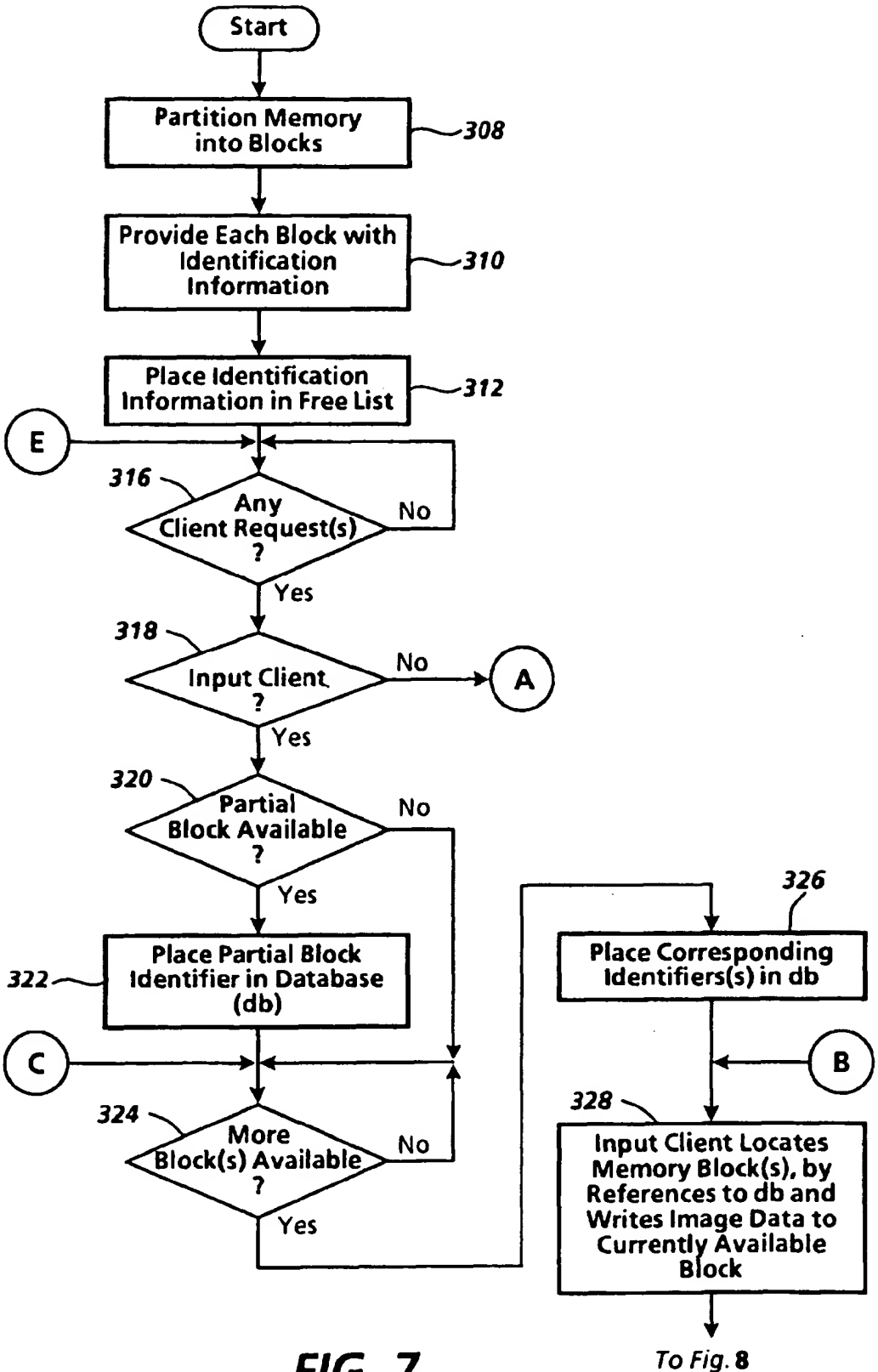
Start
Partition Memory into Blocks
308
Provide Each Block with Identification Information
310
Place Identification Information in Free List
312
E
316
Any Client Request(s) ?
No
Yes
318
Input Client ?
No
A
Yes
320
Partial Block Available ?
No
Yes
322
Place Partial Block Identifier in Database (db)
C
324
More Block(s) Available ?
No
Yes
326
Place Corresponding Identifiers(s) in db
B
328
Input Client Locates Memory Block(s), by References to db and Writes Image Data to Currently Available Block
To Fig. 8

FIG. 7

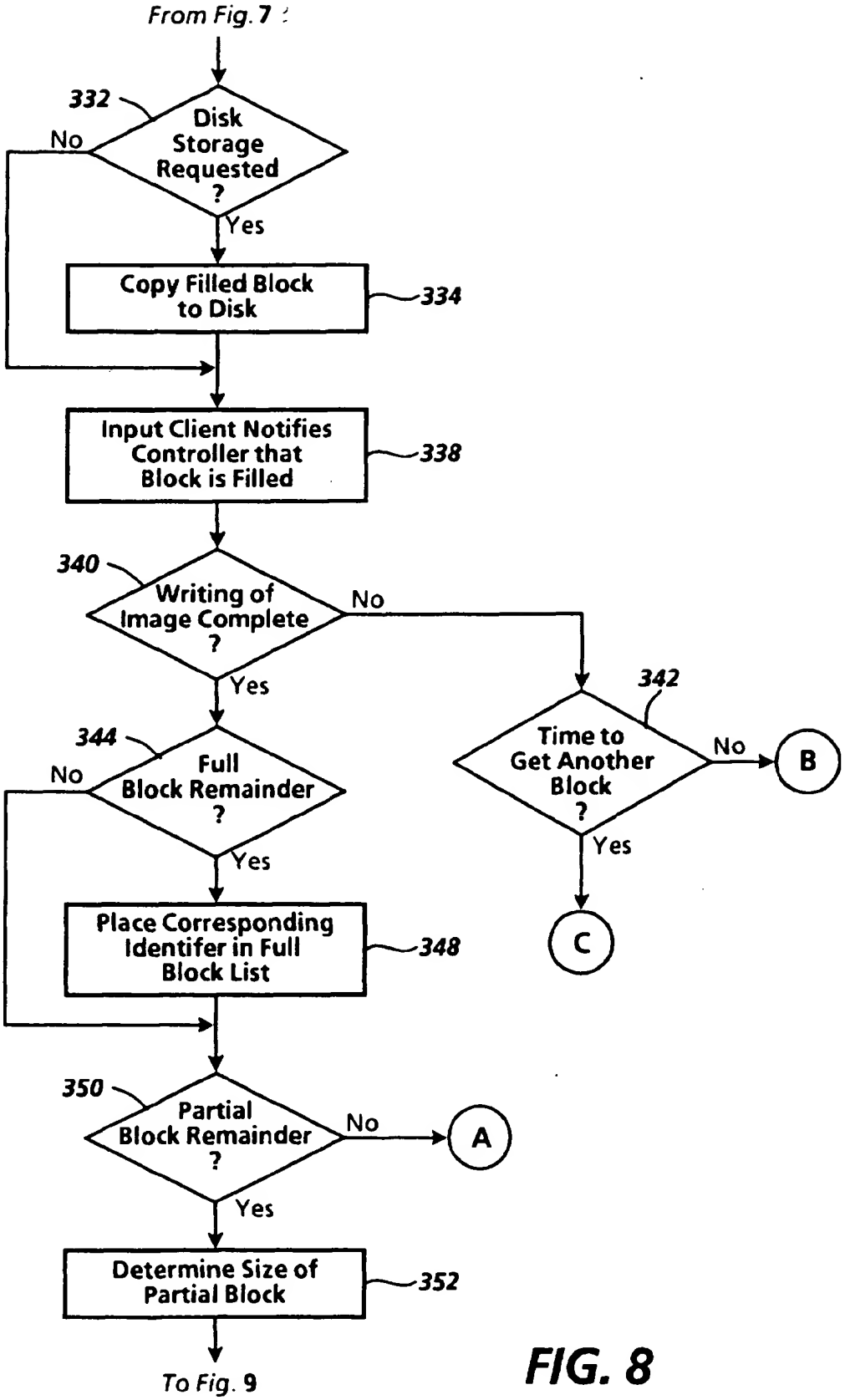
From Fig. 7
332
Disk Storage Requested ?
No
Yes
Copy Filled Block to Disk
334
Input Client Notifies Controller that Block is Filled
338
340
Writing of Image Complete ?
No
Yes
342
Time to Get Another Block ?
No
B
Yes
C
344
Full Block Remainder ?
No
Yes
Place Corresponding Identifer in Full Block List
348
350
Partial Block Remainder ?
No
A
Yes
Determine Size of Partial Block
352
To Fig. 9

FIG. 8

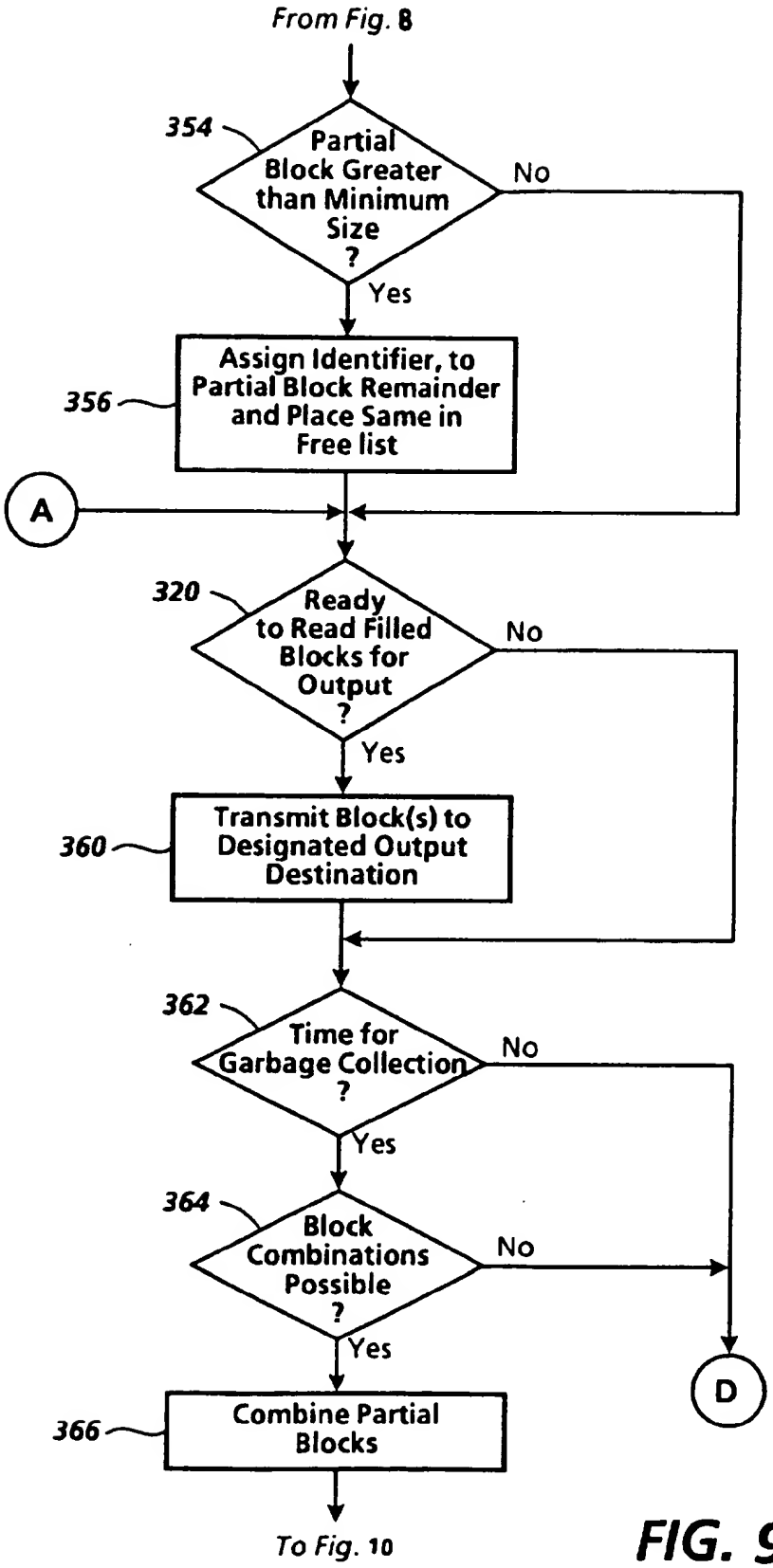
From Fig. 8
354
Partial Block Greater than Minimum Size ?
No
Yes
356
Assign Identifier, to Partial Block Remainder and Place Same in Free list
A
320
Ready to Read Filled Blocks for Output ?
No
Yes
360
Transmit Block(s) to Designated Output Destination
362
Time for Garbage Collection ?
No
Yes
364
Block Combinations Possible ?
No
Yes
D
366
Combine Partial Blocks
To Fig. 10

FIG. 9

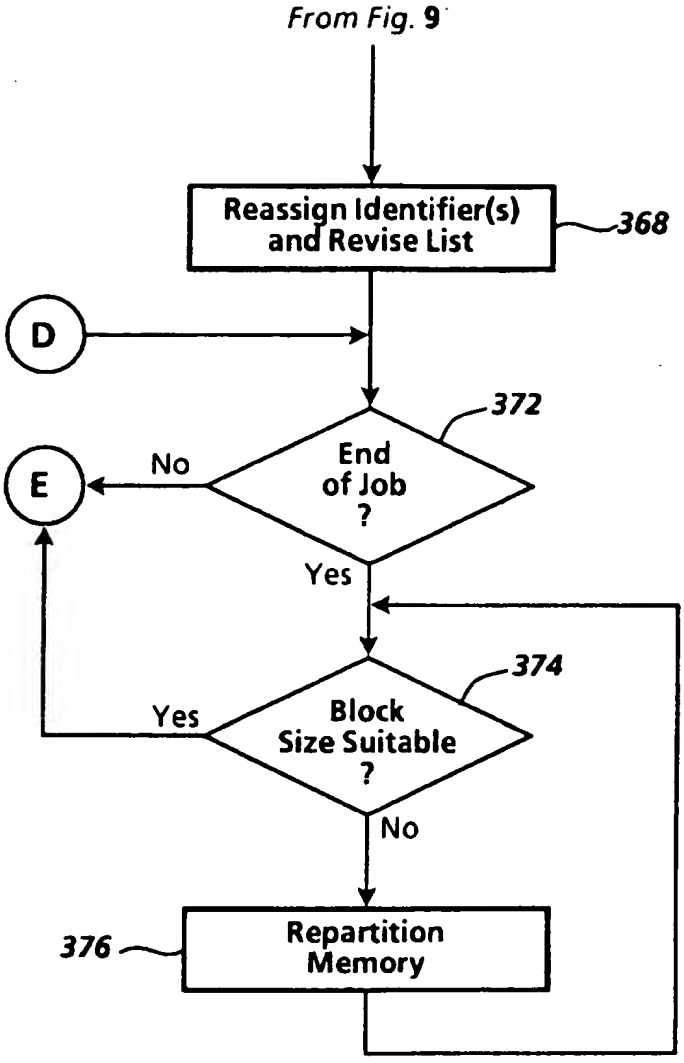
From Fig. 9
Reassign Identifier(s) and Revise List
368
D
372
End of Job ?
No
E
Yes
374
Block Size Suitable ?
Yes
No
376
Repartition Memory

FIG. 10

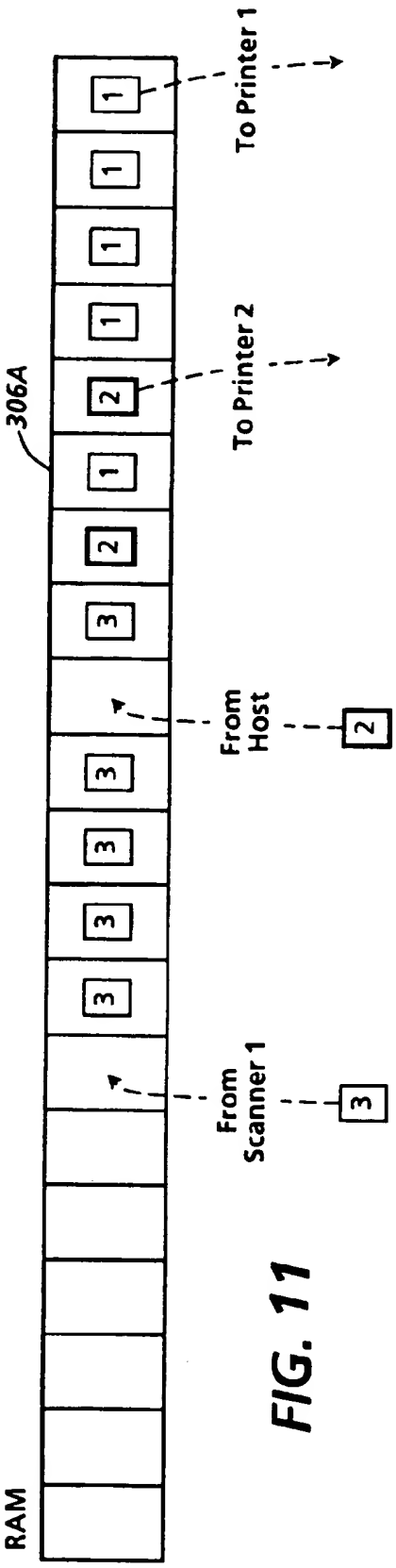
RAM
306A
From Scanner 1
From Host
To Printer 1
To Printer 2

FIG. 11

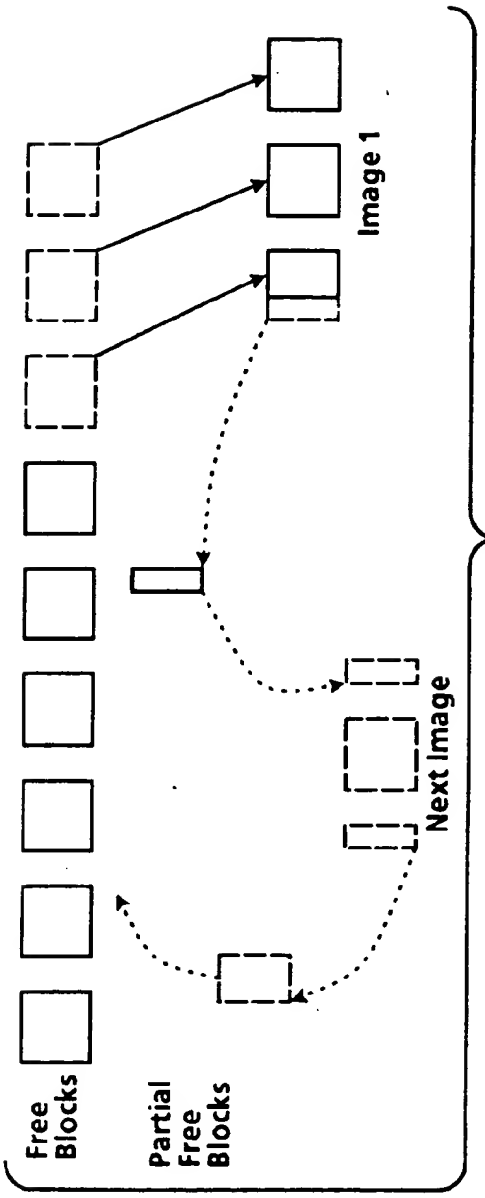
Free Blocks
Partial Free Blocks
Image 1
Next Image

FIG. 12

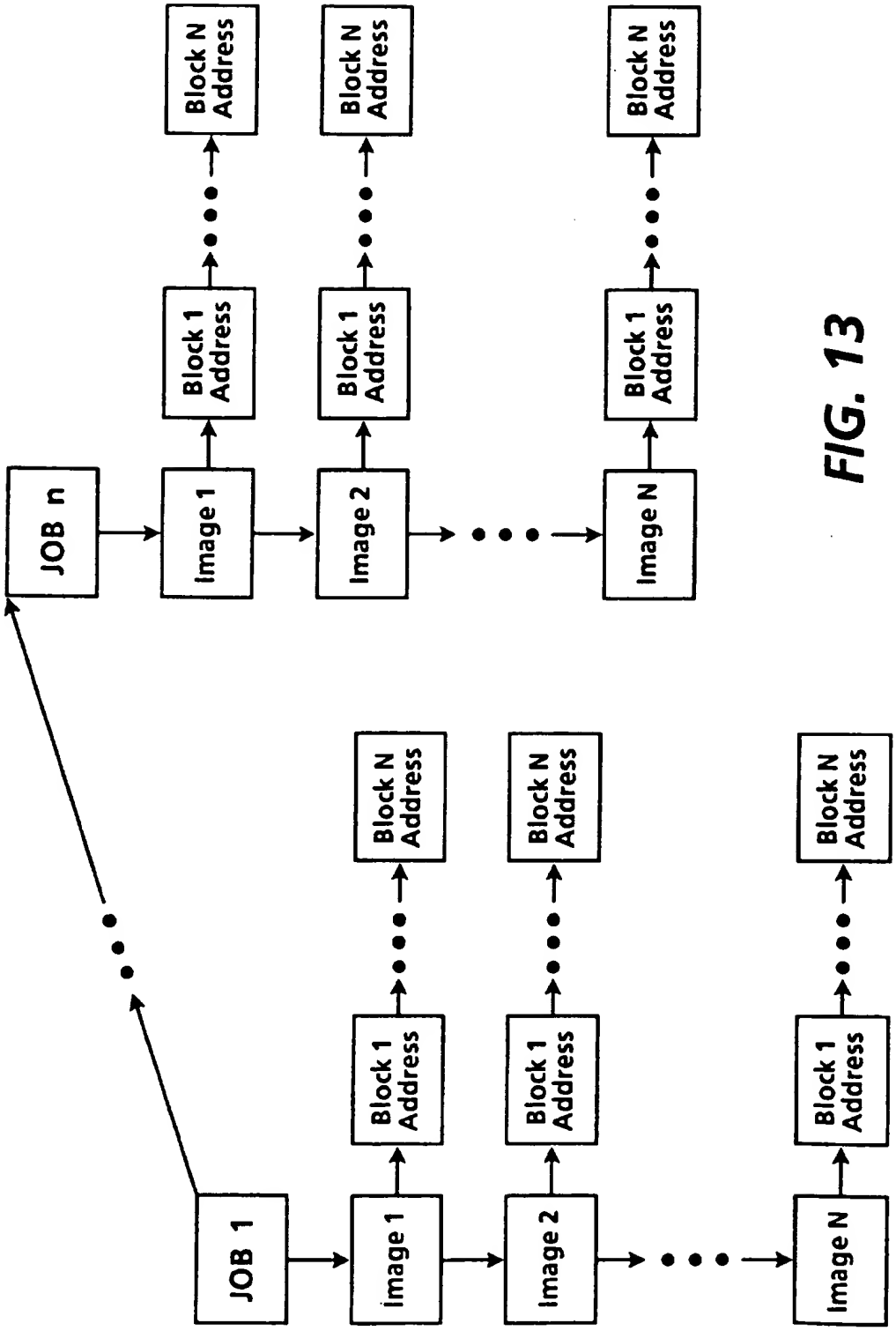
JOB n
Image 1
Image 2
Image N
Block 1 Address
Block N Address
JOB 1
Image 1
Image 2
Image N
Block 1 Address
Block N Address

FIG. 13

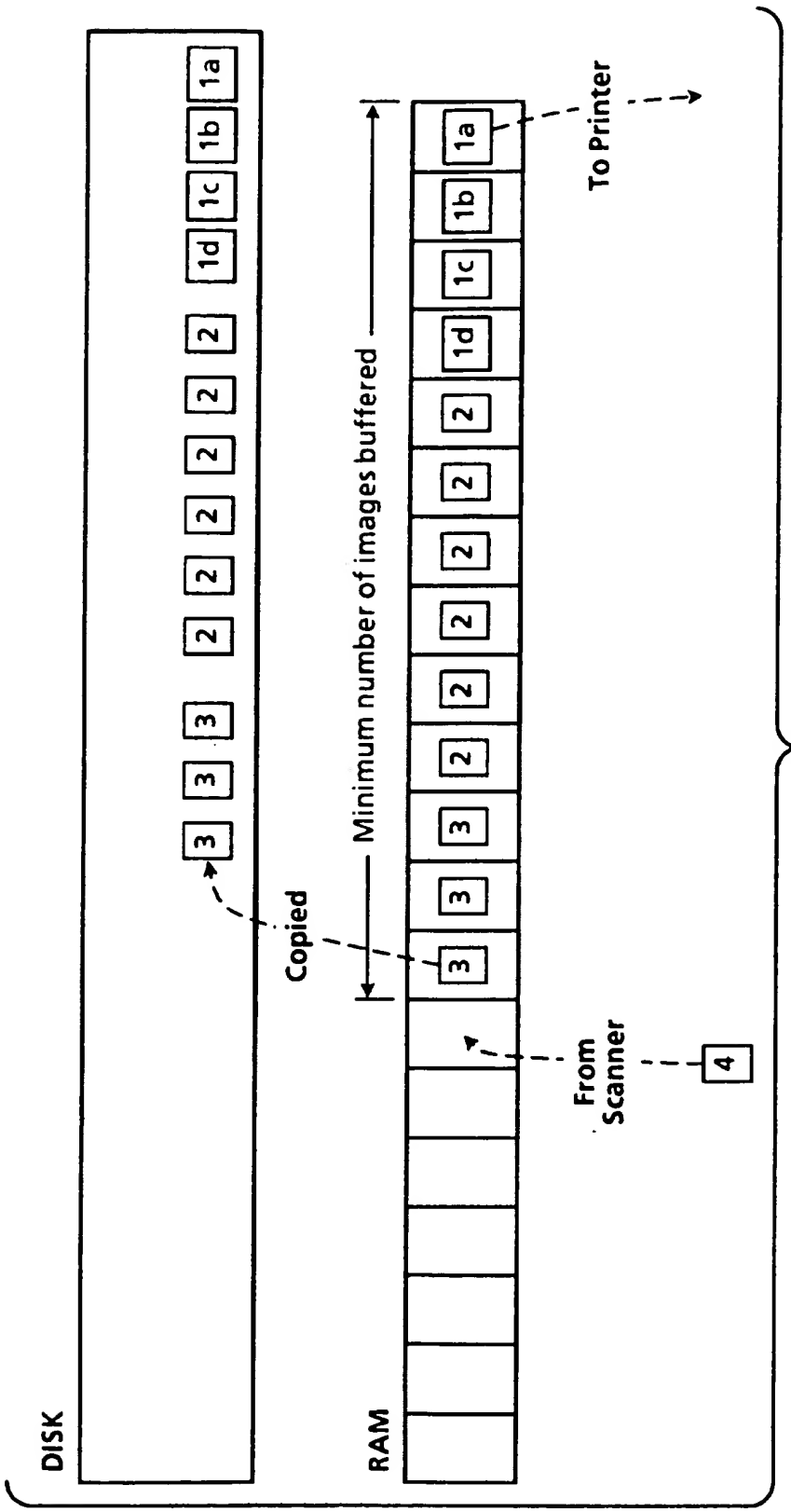
DISK
RAM
Copied
Minimum number of images buffered
From Scanner
To Printer
1a
1b
1c
1d
2
3
4

FIG. 14

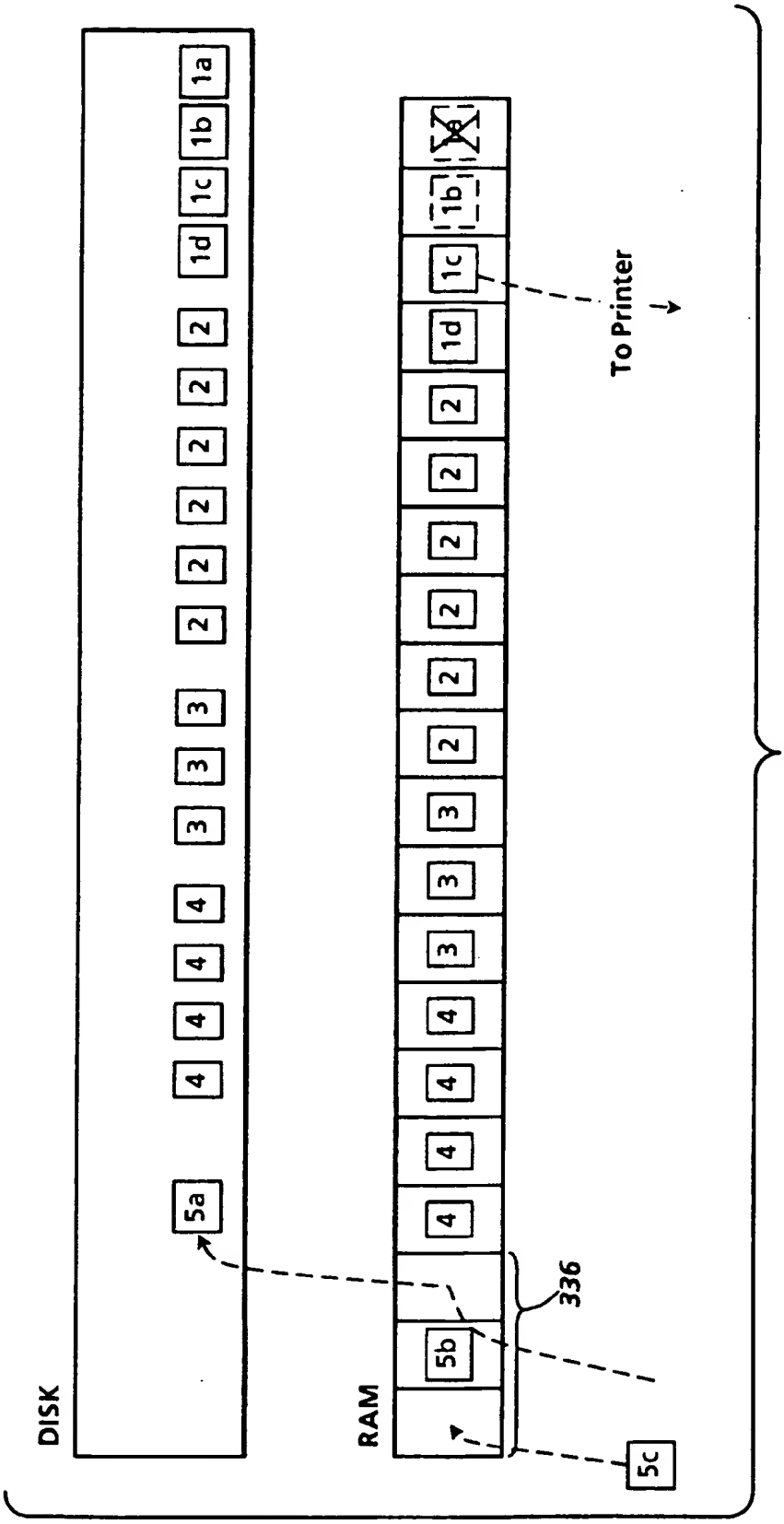

FIG. 15

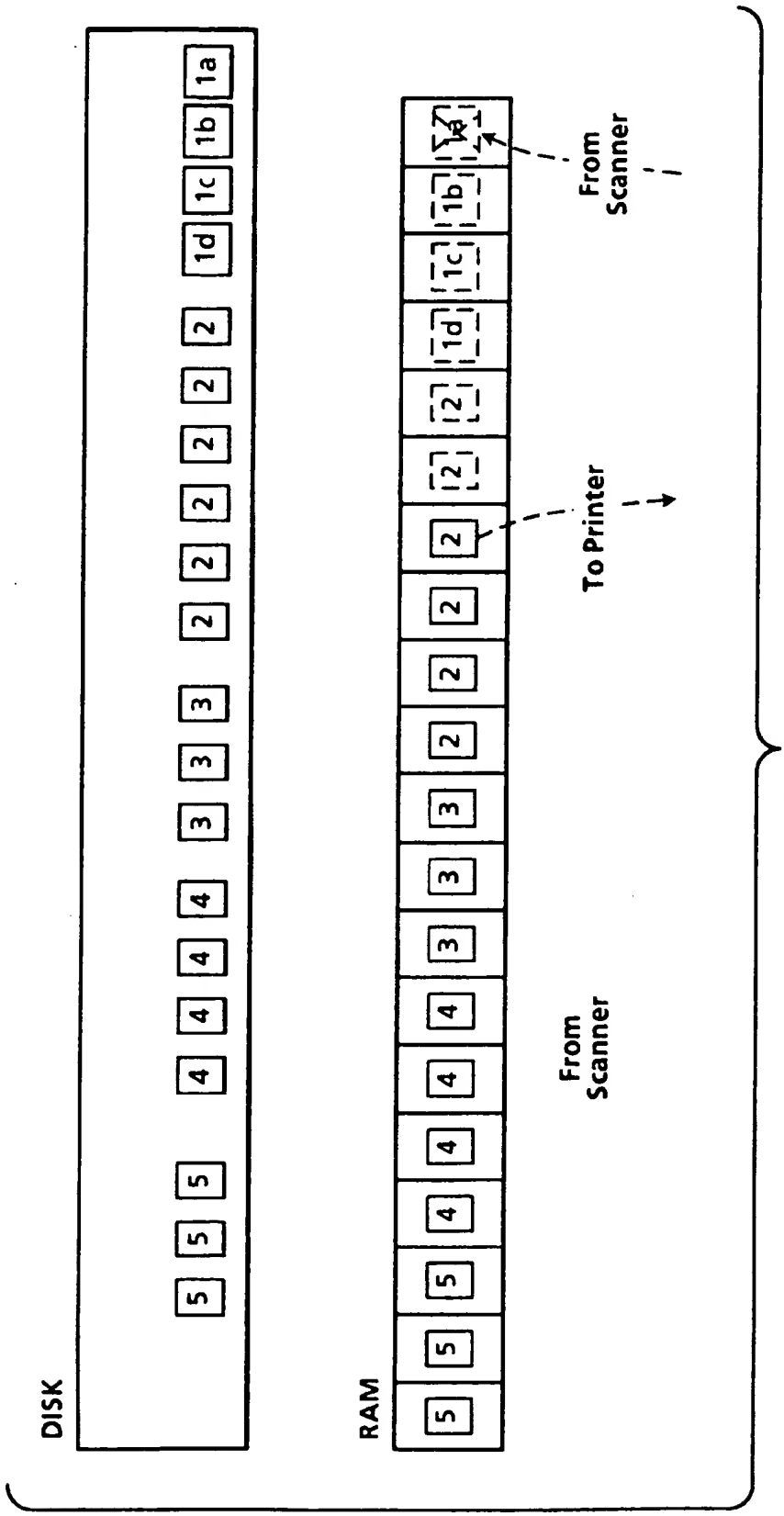
DISK
5 5 5 4 4 4 4 3 3 3 2 2 2 2 2 2 1d 1c 1b 1a
RAM
5 5 5 4 4 4 4 3 3 3 2 2 2 2
From Scanner
To Printer
From Scanner
1a 1b 1c 1d 2 2 2 2

FIG. 16

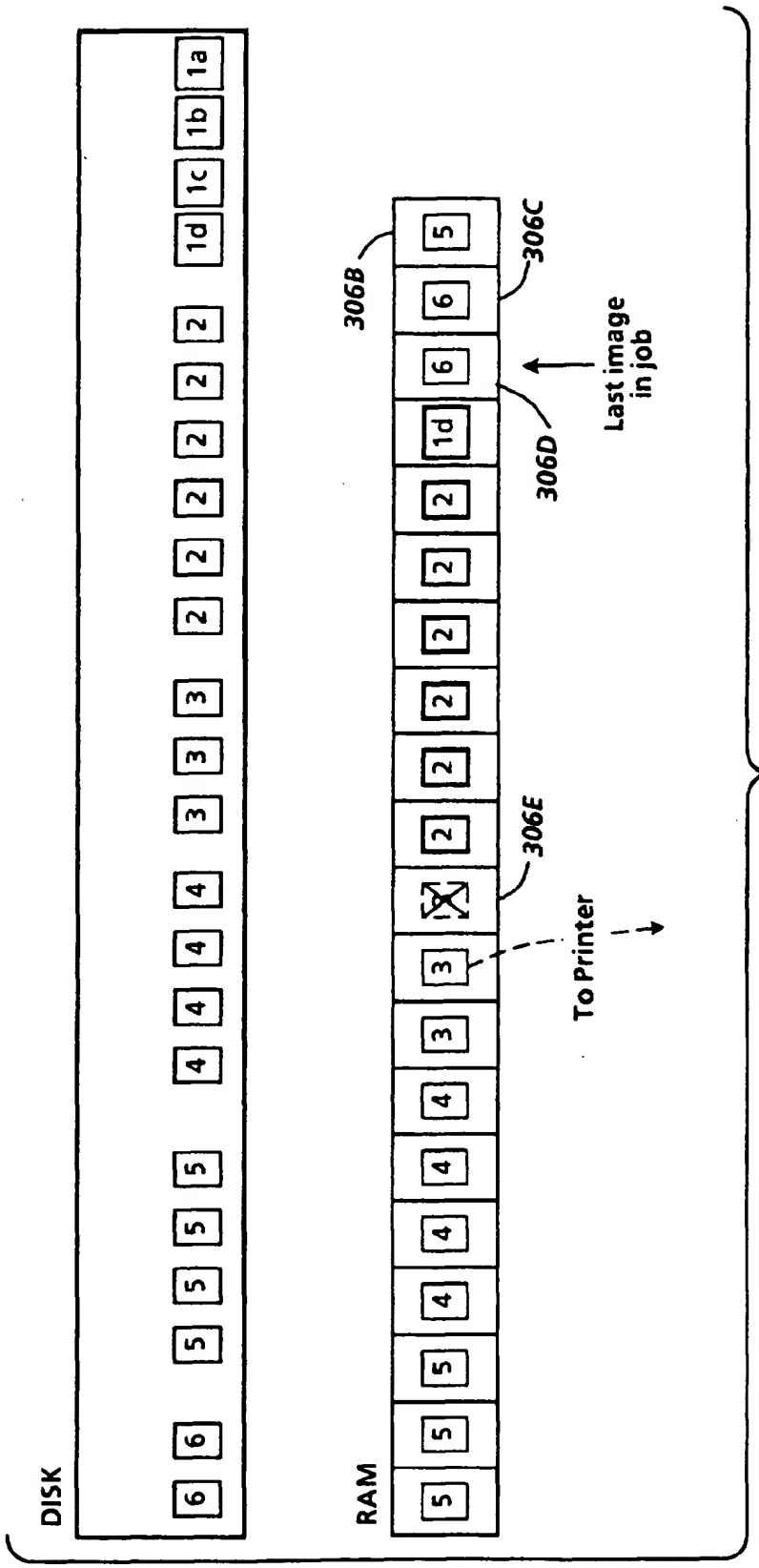
DISK
6 6 5 5 5 5 4 4 4 4 3 3 3 2 2 2 2 2 2 2 1d 1c 1b 1a
RAM
5 5 5 4 4 4 4 3 3 2 2 2 2 2 2 2 1d 6 6 5
306B
306C
306D
306E
Last image in job
To Printer

FIG. 17

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 95 30 6919

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X Y	EP-A-0 550 158 (CANON KABUSHIKI KAISHA) * figures 5,6 * * column 2, line 17 - line 34 * * column 7, line 3 - line 47 * * column 9, line 9 - column 10, line 12 *	1,4,6,7,10 2,8	G06F3/12
Y A	WO-A-91 06058 (UNISYS CORPORATION) * figures 1,12A-12C * * figures 14A-14D * * page 12, line 1 - page 13, line 17 * * page 75, line 1 - page 78, line 1 * * page 84, line 22 - line 26 *	2,8 1-8,10	
			TECHNICAL FIELDS SEARCHED (Int.Cl.6) G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	15 January 1996	Weiss, P

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 704 792 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
13.06.2001 Bulletin 2001/24

(51) Int Cl.[7]: **G06F 3/12**

(21) Application number: **95306919.2**

(22) Date of filing: **29.09.1995**

(54) Method of managing memory allocation in a printing system

Verfahren zur Verwaltung der Speicherzuweisung in einem Druckersystem

Méthode de gestion d'allocation de mémoire dans un système d'impression

(84) Designated Contracting States:
DE FR GB

(30) Priority: **29.09.1994 US 315274**

(43) Date of publication of application:
03.04.1996 Bulletin 1996/14

(73) Proprietor: **XEROX CORPORATION**
Rochester, New York 14644 (US)

(72) Inventors:
- **Ambalavanar, Samuel D.**
  Rochester NY 14625 (US)
- **Romano, Kenneth D.**
  Webster NY 14580 (US)
- **Sanford, Ronnie E.**
  Webster NY 14580 (US)
- **Frumusa, Anthony M.**
  Penfield NY 14526 (US)
- **Diaz, Orlando**
  Rochester NY 14620 (US)

(74) Representative: **Grünecker, Kinkeldey,**
Stockmair & Schwanhäusser Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(56) References cited:
EP-A- 0 550 158
WO-A-91/06058

EP 0 704 792 B1

Description

[0001] The present invention relates generally to a technique of memory management for a printing system and, more particularly, to a method of managing memory allocation for the printing system which minimizes both processing overhead and memory fragmentation.

[0002] Electronic printing systems typically include an input section, sometimes referred to as an input image terminal ("IIT"), a controller, sometimes referred to as an electronic subsystem ("ESS") and an output section or print engine, sometimes referred to as an image output terminal ("IOT"). In one type of electronic printing system, manufactured by Xerox® Corporation, known as the DocuTech® electronic printing system, a job can be inputted to the IIT from, among other sources, a network or a scanner. An example of an IIT with both network and scanner inputs is found in US-A-5,170,340.

[0003] Since digital printing systems store images electronically, a significant amount of memory is often required for storage. In a multifunctional digital printing system, various clients, i.e. various input/output devices of the printing system, seek to use the memory. That is, input clients seek to access the memory for storing image data and output clients seek to access the memory for the sake of consuming image data. Without some sort of arrangement for controlling employment of the memory by these clients, operation of the system can be impaired greatly. For example, a client with relatively slow processing capability can monopolize use of the memory at the expense of a client with relatively fast processing capability. Moreover, the memory needs of a group of clients may vary, among individual clients, over time. In the area of computer architecture, it is known that memory or resource management is a desirable approach for insuring that allocation of memory among a group of clients is performed in an orderly manner.

[0004] US-A-5,212,566 is directed toward a resource allocation scheme for a memory arrangement including disk and system memories, the system memory including a plurality of buffers. A system state controller communicates with the system memory, by way of a resource manager, and with a scanner, by way of a scan management arrangement, the scan management arrangement including a scan scheduler and a scan controller. In operation, the scanner, along with each client requesting use of the system memory, is allocated a set of buffers. During run time, the scanner fills buffers allocated to it with scanned data of a scan job, obtained by reading a document, and stores each filled buffer out to disk. If the scanner requires more buffers than are allocated to it, then a fault will occur. In response to the fault, a fault command flows from the scan controller to the system state controller, which system state controller, in turn, transmits a control command requesting the resource manager to adjust buffer allocation in the system memory. Under ideal circumstances, the scanner receives the buffers it needs to complete the scan job. As disclosed, reallocation includes obtaining a previously allocated buffer from a client other than the scanner.

[0005] While the resource management scheme of US-A-5,212,566 patent is well suited for its intended purpose, it appears to require a fair amount of processing overhead since buffer allocation among clients must often be assessed in obtaining a free buffer for the scanner. It would be desirable to provide a resource management scheme which uses a minimum amount of processing overhead in obtaining and providing memory for a given client. Additionally, in the resource management scheme of US-A-5,212,566, all of the buffers allocated to a given client may not be used fully. This can lead to underutilization of memory space and even memory fragmentation. It would be desirable to provide a resource management scheme in which all memory space of the system memory is employed in the most efficient manner possible.

[0006] Further, EP-A 0 550 158 describes a printing apparatus and printing control method wherein, when data is received from a first host computer, the printing apparatus stores the received data in a reception buffer that has been segmented into memory blocks of a prescribed size. When one memory block is filled to capacity with data received from the first host computer, the printing apparatus determines whether there is print request from another host computer. If there is a print request from another host computer, reception of printing data accompanying this request is started, and the data is stored in an used memory block of the reception buffer. When this memory block is filled to capacity, reception of data from the first host computer is resumed. The printing apparatus receives data from a plurality of host computers concurrently and prints out data the reception of which has ended. The printing apparatus is capable of deciding the order of print out based not only upon the order in which print requests occur but also upon the size of data from each host computer.

[0007] WO 91/06058 describes a storage and retrieval system for document image data. A high-capacity and high-speed storage/retrieval system provides storage and retrieval for document images in digitised data form. Clusters of storage/retrieval modules receive serialised image optical data read from documents via point-to-point controllers. The storage/retrieval modules store or exchange digital data via individual controllers or line controllers in the clusters of the storage/retrieval modules. A host computer is operative to transmit via server/controller commands and management data to remotely located storage/retrieval modules. Local workstations are connected to the storage/retrieval modules via standard interface boards and remote workstations may be connected through modems and server/controller to other remote workstations.

[0008] It is an object of the present invention to provide a resource management scheme which uses a minimum amount of processing overhead in obtaining and

providing memory for a given client and employs all memory space of the memory in the most efficient manner possible.

[0009] This object is solved by the method as claimed in independent claims 1 and 7. Preferred embodiments of the invention are subject-matters of dependent claims.

[0010] In accordance with one aspect of the present invention there is provided a method of managing memory allocation in a printing system with a controller and memory, the controller having a resource manager for managing use of the memory and the printing system supporting input clients, each input client seeking to store one or more images, in the form of image data, in the memory, comprising the steps of: partitioning at least a portion of the memory to create a plurality of blocks; providing each of the plurality of blocks with an identifier, each identifier indicating a location of a block in the memory; in response to a request from a selected input client, placing a first set of identifiers, corresponding with a first set of blocks, in a database; accessing the first set of identifiers with the selected input client; filling up one or more of the first set of blocks, with image data, by referring to the first set of identifiers; transmitting an interrupt signal to the controller, with the selected input client, each time one of the first set of blocks is filled; and in response to a designated one of the blocks being filled, placing a second set of identifiers, corresponding with a second set of blocks, in the database, so that when the selected input client has filled up a last one of the first set of blocks, the selected input client accesses the second set of blocks, by reference to the second set of identifiers and begins filling up a first block of the second set of blocks wherein memory allocation is accomplished with a minimum amount of communication between various components of the printing system.

[0011] In accordance with another aspect of the present invention there is provided a method of managing memory allocation in a printing system with an input client, the input client storing image data in memory for outputting an image associated with the image data, comprising the steps of: providing to the input client, a first set of blocks including a first block and a second block, wherein the second block includes a first part and a second part; filling up both the first block and the first part of the second block with image data, an end of the first part of the second block representing a corresponding end of a first image; and filling up the second part of the second block with image data, the image data in the second part of the second block corresponding with a second image, wherein the first image is different than the second image and usage of memory space for storing image data is maximized.

[0012] These and other aspects of the invention will become apparent from the following description, the description being used to illustrate a preferred embodiment of the invention when read in conjunction with the accompanying drawings, in which:-

Figure 1 is a block diagram depicting a multifunctional, network adaptive printing machine;

Figure 2 is a block diagram of a video control module for the printing machine of Figure 1;

Figure 3 is a block diagram of a transfer module used in conjunction with the printing machine of Figure 2;

Figure 4 is a block diagram of a facsimile card used in conjunction with the printing machine of Figure 2;

Figure 5 is a block diagram of a network controller for the printing machine of Figure 1;

Figure 6 is a block diagram of a resource management scheme including a selected number of components from the block diagram of Figure 2;

Figures 7-10 is a flow diagram illustrating some of the functionality of the resource management scheme of Figure 6;

Figure 11 is a schematic representation of electronic precollation (EPC) memory being used in conjunction with the resource management scheme of the present invention;

Figure 12 is a schematic representation illustrating how partial blocks are employed, in the resource management scheme, to reduce memory fragmentation;

Figure 13 is a schematic representation of a database format used in conjunction with the resource management scheme; and

Figures 14-17 are schematic representations illustrating how a combination of electronic precollation (EPC) and disk memory is used in conjunction with the resource management scheme.

[0013] Referring to Figure 1, a multifunctional, network adaptive printing system is designated by the numeral 10. The printing system 10 includes a printing machine 12 operatively coupled with a network service module 14. The printing machine 12 includes an electronic subsystem 16, referred to as a video control module (VCM), communicating with a scanner 18 and a printer 20. In one example, the VCM 16, which will be described in further detail below, coordinates the operation of the scanner and printer in a digital copying arrangement. In a digital copying arrangement, the scanner 18 (also referred to as image input terminal (IIT)) reads an image on an original document by using a CCD full width array and converts analog video signals, as gathered, into digital signals. In turn, an image processing system 22 (Figure 2), associated with the scanner 18, executes signal correction and the like, converts the corrected signals into multi-level signals (e.g. binary signals), compresses the multi-level signals and preferably stores the same in electronic precollation (EPC) memory 24.

[0014] Referring again to Figure 1, the printer 20 (also referred to as image output terminal (IOT)) preferably includes a xerographic print engine. In one example, the print engine has a multi-pitch belt (not shown) which is

written on with an imaging source, such as a synchronous source (e.g. laser raster output scanning device) or an asynchronous source (e.g. LED print bar). In a printing context, the multi-level image data is read out of the EPC memory 24 (Figure 2) while the imaging source is turned on and off, in accordance with the image data, forming a latent image on the photoreceptor. In turn, the latent image is developed with, for example, a hybrid jumping development technique and transferred to a print media sheet. Upon fusing the resulting print, it may be inverted for duplexing or simply outputted. It will be appreciated by those skilled in the art that the printer can assume other forms besides a xerographic print engine without altering the concept upon which the disclosed embodiment is based. For example, the printing system 10 could be implemented with a thermal ink jet or ionographic printer.

[0015] Referring specifically to Figure 2, the VCM 16 is discussed in further detail. The VCM 16 includes a video bus (VBus) 28 with which various I/O, data transfer and storage components communicate. Preferably, the VBus is a high speed, 32 bit data burst transfer bus which is expandable to 64 bit. The 32 bit implementation has a sustainable maximum bandwidth of approximately 60 MBytes/sec. In one example, the bandwidth of the VBus is as high as 100 MBytes/sec.

[0016] The storage components of the VCM reside in the EPC memory section 30 and the mass memory section 32. The EPC memory section includes the EPC memory 24, the EPC memory being coupled with the VBus by way of a DRAM controller 33. The EPC memory, which is preferably DRAM, provides expansion of up to 64 MBytes, by way of two high density 32 bit SIMM modules. The mass memory section 32 includes a SCSI hard drive device 34 coupled to the VBus by way of a transfer module 36a. As will appear, other I/O and processing components are coupled respectively to the VBus by way of transfer modules 36. It will be appreciated that other devices (e.g. a workstation) could be coupled to the VBus by way of the transfer module 36a through use of a suitable interface and a SCSI line.

[0017] Referring to Figure 3, the structure of one of the transfer modules 36 is discussed in further detail. The illustrated transfer module of Figure 3 includes a packet buffer 38, a VBus interface 40 and DMA transfer unit 42 . The transfer module 36, which was designed with "VHSIC" Hardware Description Language (VHDL), is a programmable arrangement permitting packets of image data to be transmitted along the VBus at a relatively high transfer rate. In particular, the packet buffer is programmable so that the segment or packet can be varied according to the available bandwidth of the VBus. In one example, the packet buffer can programmed to handle packets of up to 64 Bytes Preferably, the packet size would be reduced for times when the VBus is relatively busy and increased for times when activity on the bus is relatively low.

[0018] Adjustment of the packet size is achieved with the VBus interface 40 and a system controller 44 (Figure 5). Essentially, the VBus interface is an arrangement of logical components, including, among others, address counters, decoders and state machines, which provides the transfer module with a selected degree of intelligence. The interface 40 communicates with the system controller to keep track of desired packet size and, in turn, this knowledge is used to adjust the packet size of the packet buffer 38, in accordance with bus conditions. That is, the controller, in view of its knowledge regarding conditions on the VBus 28, passes directives to the interface 40 so that the interface can adjust packet size accordingly. Further discussion regarding operation of the transfer module 36 is provided below

[0019] More particularly, each DMA transfer unit employs a conventional DMA transfer strategy to transfer the packets. In other words, the beginning and end addresses of the packet are used by the transfer unit in implementing a given transfer. When a transfer is complete, the interface 40 transmits a signal back to the system controller 44 so that further information, such as desired packet size and address designations, can be obtained.

[0020] Referring to Figures 1 and 2, three I/O components are shown as being coupled operatively to the VBus 28, namely a FAX module 48, the scanner or IIT 18, and the printer or IOT 20; however, it should be recognized that a wide variety of components could be coupled to the VBus by way an expansion slot 50. Referring to Figure 4, an implementation for the FAX module, which is coupled to the VBus 28 by way of transfer module 36b, is discussed in further detail. In the preferred embodiment, a facsimile device (FAX) 51 includes a chain of components, namely a section 52 for performing Xerox adaptive compression/decompression, a section 54 for scaling compressed image data, a section 56 for converting compressed image data to or from CCITT format, and a modem 58, preferably manufactured by Rockwell Corporation, for transmitting CCITT formatted data from or to a telephone, by way of a conventional communication line.

[0021] Referring still to Figure 4, each of the sections 52, 54 and 56 as well as modem 58 are coupled with the transfer module 36b by way of a control line 60. This permits transfers to be made to and from the FAX module 48 without involving a processor. As should be understood, the transfer module 36b can serve as a master or slave for the FAX module in that the transfer module can provide image data to the FAX for purposes of transmission or receive an incoming FAX. In operation, the transfer module 36b reacts to the FAX module in the same manner that it would react to any other I/O component. For example, to transmit a FAX job, the transfer module 36b feeds packets to the section 52 through use of the DMA transfer unit 42 and, once a packet is fed, the transfer module transmits an interrupt signal to the system processor 44 requesting another packet. In one embodiment, two packets are maintained in the packet

buffer 38 so that "ping-ponging" can occur between the two packets. In this way, the transfer module 36b does not run out of image data even when the controller cannot get back to it immediately upon receiving an interrupt signal.

[0022] Referring again to Figure 2, the IIT 18 and IOT 20 are operatively coupled to the VBus 28 by way of transfer modules 36c and 36d. Additionally, the IIT 18 and the IOT 20 are operatively coupled with a compressor 62 and a decompressor 64, respectively. The compressor and decompressor are preferably provided by way of a single module that employs Xerox adaptive compression devices. Xerox adaptive compression devices have been used for compression/decompression operations by Xerox Corporation in its DocuTech® printing system. In practice, at least some of the functionality of the transfer modules is provided by way of a 3 channel DVMA device, which device provides local arbitration for the compression/decompression module.

[0023] As further illustrated by Figure 2, the scanner 18, which includes the image processing section 22, is coupled with an annotate/merge module 66. Preferably the image processing section includes one or more dedicated processors programmed to perform various desired functions, such as image enhancement, thresholding/screening, rotation, resolution conversion and TRC adjustment. The selective activation of each of these functions can be coordinated by a group of image processing control registers, the registers being programmed by the system controller 44. Preferably, the functions are arranged along a "pipeline" in which image data is inputted to one end of the pipe, and image processed image data is outputted at the other end of the pipe. To facilitate throughput, transfer module 36e is positioned at one end of the image processing section 22 and transfer module 36c is positioned at another end of the section 22. As will appear, positioning of transfer modules 36c and 36e in this manner greatly facilitates the concurrency of a loopback process.

[0024] Referring still to Figure 2, arbitration of the various bus masters of the VCM 16 is implemented by way of a VBus arbiter 70 disposed in a VBus arbiter/bus gateway 71. The arbiter determines which bus master (e.g. FAX module, Scanner, Printer, SCSI Hard Drive, EPC Memory or Network Service Component) can access the VBus at one given time. The arbiter is made up of two main sections and a third control section. The first section, i.e., the "Hi-Pass" section, receives input bus requests and current priority selection, and outputs a grant corresponding to the highest priority request pending. The current priority selection input is the output from the second section of the arbiter and is referred to as "Priority Select". This section implements priority rotation and selection algorithm. At any given moment, the output of the logic for priority select determines the order in which pending requests will be serviced. The input to Priority Select is a register which holds an initial placement of devices on a priority chain. On servicing requests, this logic moves the devices up and down the priority chain thereby selecting the position of a device's next request. Control logic synchronizes the tasks of the Hi-Pass and the Priority Select by monitoring signals regarding request/grant activity. It also prevents the possibility of race conditions.

[0025] Referring to Figure 5, the network service module 14 is discussed in further detail. As will be recognized by those skilled in the art, the architecture of the network service module is similar to that of a known "PC clone". More particularly, in the preferred embodiment, the controller 44, which preferably assumes the form of a SPARC processor, manufactured by Sun Microsystems, Inc., is coupled with a standard SBus 72. In the illustrated embodiment of Figure 5, a host memory 74, which preferably assumes the form of DRAM, and a SCSI disk drive device 76 are coupled operatively to the SBus 72. While not shown in Figure 5, a storage or I/O device could be coupled with the SBus with a suitable interface chip. As further shown in Figure 5, the SBus is coupled with a network 78 by way of an appropriate network interface 80. In one example, the network interface includes all of the hardware and software necessary to relate the hardware/software components of the controller 44 with the hardware/software components of the network 78. For instance, to interface various protocols between the network service module 14 and the network 78, the network interface could be provided with, among other software, Netware® from Novell Corp.

[0026] In one example, the network 78 includes a client, such as a workstation 82 with an emitter or driver 84. In operation, a user may generate a job including a plurality of electronic pages and a set of processing instructions. In turn, the job is converted, with the emitter, into a representation written in a page description language, such as PostScript. The job is then transmitted to the controller 44 where it is interpreted with a decomposer, such as one provided by Adobe Corporation.

[0027] Referring again to Figure 2, the network service module 14 is coupled with the VCM 16 via a bus gateway 88 of the VBus arbiter/bus gateway 71. In one example, the bus gateway comprises a field programmable gate array provided by XILINX corporation. The bus gateway device provides the interface between the host SBus and the VCM VBus. It provides VBus address translation for accesses to address spaces in the VBus real address range, and passes a virtual address to the host SBus for virtual addresses in the host address range. A DMA channel for memory to memory transfers is also implemented in the bus gateway. Among other things, the bus gateway provides seamless access between the VBus and SBus, and decodes virtual addresses from bus masters, such as one of the transfer modules 36, so that an identifier can be obtained from a corresponding slave component. It will be appreciated by those skilled in the art that many components of the printing system 10 are implemented in the form of a single ASIC.

[0028] Referring to Figures 2, 3 and 5, further discussion regarding DMA transfer of each of the transfer modules 36 is provided. In particular, in one example, the images of a job are stored in the host memory 74 as a series of blocks. Referring to Figure 16, a series of blocks is shown as being stored in the EPC memory 24. Preferably, each block comprises a plurality of packets. In operation, one of the transfer modules 36 is provided, by the controller 44, with the beginning address of a block and the size of the block. In turn, for that block, the transfer module 36 effects a packet tranfer and increments/decrements a counter. This procedure is repeated for each packet of the block until the interface 40 determines, by reference to the counter, that the last packet of the block has been transferred. Typically, for each stored image, several blocks are transferred, in a packet-by-packet manner, as described immediately above.

[0029] Referring to Figure 6 a scheme for managing memory allocation, i.e. a resource management scheme, is illustrated. More particularly, the controller 44 includes a resource manager 300, while the host memory 74 includes a pair of lists 302, referred to respectively as the "free block list" and the "free partial block list", and a database ("db") 304. The resource manager is implemented by way of suitable algorithms, the details of which will be discussed in further detail below, and the significance of the lists and the database, relative to the resource management scheme, will also be discussed below. Additionally, the EPC memory 24 and the SCSI hard drive ("disk") 34 are shown as being comprised of blocks 306. A discussion of a methodology for forming and allocating memory blocks follows:

[0030] Referring to Figures 6-10, the algorithms for implementing the resource management scheme are discussed. Initially, at step 308, the EPC memory 24 is partitioned into a series of the blocks 306. A partitioned set of memory blocks is also shown in Figure 11. Preferably, the block size is varied in accordance with factors, such as image size to be stored. For example, if a location generally copies complex documents which results in poorly compressed (large) image, the block size can be increased. As will appear from the discussion below, increased block size will result in fewer interrupts by a client (e.g. scanner 18) of the controller 44.

[0031] Each block is then provided with identification information (step 310), such as block ID, block address and block size, which identification information is placed, at step 312, in the free block list. Preferably, each list in the host memory 74 is a linked list of structures. At step 316, the resource management system waits for a memory request from a client. In the present context, a client is an input or output device encompassed by the printing system 10 (Figures 1,2 and 5). A client initiates a request by transmitting a suitable request or interrupt signal to the controller 44. Upon receiving a request signal, the controller determines, via step 316, whether the client is an input client. If the client is an input client, then the process proceeds to step 318, otherwise the process proceeds directly to step 320 (Figure 9) where an output client request is serviced.

[0032] Assuming the requesting client is an input client, the resource manager 300 examines the free partial block list to determine if a partial block is available for the requesting client. Referring to Figure 12, an example of the allocation of a partial block to the beginning of an image will be discussed In particular, at system initialization, no partial block is available for an image 1 of a job. After image data for image 1 is delivered to the memory, however, a partial, unfilled block may remain. As shown in Figure 12, and explained in further detail below, the partially unfilled block, with its corresponding identifier is made available for use with the next image.

[0033] Returning to Figure 7, if a partial, unfilled block is available, then it is designated with an identifier and, at step 322, placed in the db 304. Next, at step 324, the resource manager consults the free block list to determine if a nominal number of blocks are available for use by the input client. In the preferred embodiment, each client is assigned a value corresponding to the number of nominal blocks to which it is entitled. In one example, assignment is based on the processing speed of the requesting client. That is, per each request, it may be desirable to provide fast processing clients with more blocks than slow processing clients. In one situation, the nominal number of blocks to be assigned a requesting client may not be available in the free block list. In this situation, the resource manager may provide the requesting client with one or more partial blocks until a whole block becomes available.

[0034] Assuming the nominal number of blocks is available, at step 326 the resource manager will place appropriate identifiers (i.e. information identifying both a first address and a size of each block) in the db 304. Referring to Figure 13, a suitable database structure for use with the disclosed embodiment is shown. The database is constructed in a hierarchical scheme in which jobs are linked to images and images are linked to blocks. In one example, where the client's storable image data is associated with a first image (i.e. "Image 1) of a first job (i.e. "JOB 1"), then the first block identifier is placed at the location designated as "Block 1 Address". Subsequently, the client will access the database and, at step 328 (Figure 7), locate the address of the first available block. The client will then, in cooperation with, for example, one of transfer modules 36, fill up the located block. When the scanner is serving as the client, the scanner will initiate a DMA transfer, with EPC memory 24, via the transfer module 36D (Figure 2). Referring again to Figure 14, the scanner is shown as using the EPC memory in conjunction with other clients. While the block 306A is shown as being a whole block, it will be understood that, in many instances, it would be a partial block.

[0035] The printing system 10 offers the advantageous feature of storing jobs, intended to be outputted

as multiple sets, on disk. In this way EPC memory can be made available to multiple clients in a relatively short time interval. Referring to step 332 of Figure 8, when disk storage is desired, each stored block is copied to disk 34 (also see step 334 of Figure 8). Referring to Figure 14, a graphic representation demonstrating the relationship between EPC memory and disk is provided.

[0036] As further shown in Figure 14, preferably, a minimum amount of input image pages, intended for printing, are buffered prior to printing. This has been found to be advantageous since a printer typically processes image data at a rate much greater than that of most input clients, such as the scanner. In the illustrated embodiment of Figure 15, a variable buffer zone 336 is maintained for the scan client. This buffer zone is used to move image directly to disk, which enables the system to continue scanning without stopping. It will be appreciated that the variable zone can be used by clients, other than the scanner 18, to facilitate storage.

[0037] Referring again to Figure 8, the input client transmits an interrupt signal to the controller, at step 338, when a block has been filled with image data. Alternatively, the input client could be provided, in advance, with pointers to lists of block addresses. In this way, the input client would read, without controller intervention, the locations of blocks to be used.

[0038] A determination is made at step 340 as to whether a full image has been written into EPC memory 24. Assuming that the end of the image has not been reached, it is determined, at step 342, whether another nominal number of blocks is required. It should be appreciated that, typically, when a client requires a nominal number of blocks, the resource manager provides it with a set of plural blocks. In application, those blocks follow a sequence and one of the blocks in the sequence is identified as a "relative last block" which, when reached, indicates that another set of blocks is required. The position of the relative last block is variable in that it need not, in absolute terms, be the last block of the set. if the relative last block has not been reached, then the process loops back to step 328 where the db 304 (Figure 6) is accessed so that the client can locate the next block to be filled. On the other hand, if the relative last block has been reached, then the process loops back to step 324 for obtaining at least a part of another block set.

[0039] Referring still to Figure 8, if it is determined, at step 340, that a full image has just been written into memory, then a series of steps is performed to prepare for the receiving of another image. First, the resource manager 300 (Figure 6) determines, with step 344, if all of the full blocks have been used by the input client. If not, then the identifier of each surplus whole block is placed in the free block list (step 348), otherwise the process proceeds to step 350 where the resource manager determines if the image ends on a partial block. Referring again to Figure 12, an example of how an image might end at a partial block is shown for the "Image 1". In the preferred embodiment, the size of the unused part of Image 1 is then determined in accordance with step 352 of Figure 8. Referring to Figure 9, if the size of the partial block is greater than a selected minimum size (step 354), then an identifier is assigned to the partial block (step 356) and placed in the free partial block list so that the partial block can be used to receive image data from another image, such as the "Next Image" of Figure 12. For those cases in which a given partial block is smaller than a selected minimum, the given partial block is saved for "garbage collection", the significance of which will be described below.

[0040] At steps 320 and 360, the preferred methodology accommodates for the needs of an output client, such as a printer. Regarding step 360, the output client is preferably "told" where the image data, intended for use in outputting, resides. In this way, the output client can read the image data from the EPC memory. Additionally, as shown in Figure 14, an output operation can be executed just before or after an input operation.

[0041] Referring to Figures 14, 16 and 17, an application of the the present memory management scheme, with respect to the printing client, is discussed in further detail. In the illustrated embodiments of Figures 14, 16 and 17, a given job, intended to be printed in multiple sets, is shown as including six images. In Figure 14, the first three images are buffered and copied to disk. In Figure 16, writing of images, to memory, continues concurrent with the reading of first and second image blocks by the printer. While the read/write operations are not "concurrent", in absolute terms, they appear, to a system operator, as being concurrent.

[0042] In Figure 17, the end of the job is written into EPC memory at blocks 306B, 306C and 306D, while the beginning of the printing of a second set is initiated at block 306E. For the printing of the second set, the image 2, along with the block for 1D need not be copied from disk. As should be recognized, the EPC memory and disk function in a manner comparable to a ring buffering arrangement in that image data from disk can be written over image data in the EPC memory, continually, in order to form a desired number of sets.

[0043] At step 362 (Figure 9), it is determined whether an appropriate time has arrived for "garbage collection". In the present context, garbage collection refers to combining "spent blocks", i.e. blocks having image data already "consumed" by an output client, for future use. In one example, a check for garbage collection is performed after a predesignated number of images have been printed. More particularly, garbage collection is performed as a background task, i.e. during a noncritical time of a job cycle.

[0044] To implement garbage collection, the possibility of block combination is checked at step 364 and partial blocks are combined, if possible, at step 366. It follows that block combination constitutes, in one example, linking partial blocks with references. As blocks are formed from partial blocks (step 368), some partial block identifiers will be discarded and the resulting whole

block will be placed in the free block list. If garbage collection is not performable, the process proceeds to step 372.

[0045] At step 372, a check is performed to determine if the currently completed image is the last image in the job. If the image is not the last image, then the process loops back to step 316 where the input/output client accesses the db 304 for another block identifier, assuming that the client is ready. If, on the other hand, the job is complete, then a determination is made, at step 374, as to whether repartitioning is required. Repartitioning is performed (step 376) until a suitable block size is obtained. Subsequent to repartitioning the process loops back to step 316.

[0046] Numerous features of the above-disclosed embodiment will be appreciated by those skilled in the art:

[0047] First, the disclosed resource management scheme functions with a minimum amount of processing overhead. In particular, per a request by a client, block identifiers are placed at selected locations in a database. In turn, a selected client can access the database, determine where available blocks exist in memory, by reference to the block identifiers, and begin filling those blocks. As the client fills the blocks, it signals a controller and, in response to the signals, the controller indicates to a resource manager when to place more block identifiers in the database for the selected client. The process of providing blocks is transparent to the selected client and the resource manager is not required to possesses any significant knowledge about memory allocations of system clients in order to service the selected client appropriately.

[0048] Second, memory utilization of the system is enhanced in that memory blocks are employed in a particularly efficient manner. That is, unused blocks, whether they be partial or whole blocks, are placed in one or more free lists as soon as a given client ceases to have an immediate need for them. In turn, unused free blocks are allocated to other clients, in an expeditious manner and partial blocks are, in many instances, used to store the beginning and end parts of an image. In this way memory fragmentation is minimized and memory space is made available to clients, who have an urgent demand for it, as soon as possible.

[0049] Third, the resource management technique is flexible in that a wide variety of system parameters, such as nominal block size, number of blocks allocated to a given client at one time, and block allocation timing are variable. In this way, the attendant printing system can accommodate for varying input/output demands. Preferably, the system can keep track of compression ratios and adjust the variables to maximize performance for a particular location's majority usage. For example, if a location generally copies complex documents, which results in poorly compressed (large) images, the block size can be increased. Use of increased block size will result in fewer interrupts to the controller. Moreover, the system can, among other things, adjust block allocation according to individual client processing capability and predict the moment at which blocks, for a given client, should be made available.

[0050] Finally, the resource management scheme can be used to coincidentally manage volatile and nonvolatile memory in a manner that maximizes the functionality of the volatile memory, which volatile memory may be limited in space. Through efficient management of volatile and nonvolatile memory, even a job, having a size greater than that of the volatile memory (e.g. EPC memory) can be outputted in multiple sets with relatively little degradation in output rate. Moreover, volatile and nonvolatile memory can be used, conjunctively, to insure that even complex documents, having much greater size than that of the volatile memory, can be stored by a given input client without impairing operation of that given input client.

Claims

1. A method of managing memory allocation in a printing system (10) comprising a controller (44) and memory (24, 34), the controller (44) having a resource manager (300) for managing use of the memory (24, 34), at least a portion of the memory (24, 34) being partitioned into a plurality of blocks (306), the printing system (10) supporting input clients (18), each input client (18) seeking to store one or more images, in the form of image data, in the memory (24, 34),
characterized by:

providing each block (306) with an identifier indicating a location of the block (306) in the memory (24, 34);

in response to a request from one of the input clients (18), placing a first set of identifiers, corresponding with a first set of blocks (306), in a database (304);

accessing the first set of identifiers with the input client (18);

filling one or more blocks (306) of the first set of blocks (306), with image data from the input client (18), by referring to the first set of identifiers;

transmitting an interrupt signal with the input client (18) to the controller (44), each time a block (306) of the first set of blocks (306) is filled; and

in response to receiving the interrupt signal at the controller (44), placing a second set of identifiers, corresponding with a second set of

blocks (306), in the database (304), in case that a predetermined block (306) of the first set of blocks (306) has been filled, wherein the input client (18), after filling a last block (306) of the first set of blocks (306), is enabled to access the second set of blocks (306), by referring to the second set of identifiers, and to fill blocks (306) of the second set of blocks (306), thereby accomplishing memory allocation with a minimum amount of communication within the printing system (10).

2. The method of claim 1 wherein said step of filling includes filling the first set of blocks (306) with image data generated from a scan service, the scan service including a scanning apparatus (18) for converting information, disposed on a document, to the image data.

3. The method of claim 1 or 2 wherein:

the first set of blocks (306) includes a last block; and

said step of placing the second set of identifiers includes placing the second set of identifiers in the database (304) after the last block has been filled with image data.

4. The method of one of claims 1 to 3, further comprising:

outputting a corresponding job, including one or both of the first and second sets of blocks (306), with an output client (20).

5. The method of one of claims 1 to 4 wherein all of a filled block is used to print part of a corresponding job, further comprising:

placing an identifier, associated with the used, filled block, in a first list (302).

6. The method of one of claims 1 to 5, further comprising:

repartitioning the memory (24, 34), after a predetermined number of blocks (306) have been filled, for changing block size in accordance with varying image size of individual images being stored in memory (24, 34).

7. A method of managing memory allocation in a printing system (10) comprising memory (24, 34), at least a portion of the memory (24, 34) being partitioned into a plurality of blocks (306), the printing system (10) supporting input clients (18), an input client (18) storing a first image and a second image, in the form of image data, in the memory (24, 34), **characterized by**:

providing the input client (18) with a first set of blocks (306) including a first block and a second block, the second block including a first part and a second part;

filling both the first block and the first part of the second block with image data of the first image, an end of the first part of the second block representing a corresponding end of the first image; and

filling the second part of the second block with image data of the second image, wherein the first image is different than the second image, thereby maximizing usage of memory space for storing image data.

8. The method of claim 7, further comprising:

designating the second part of the second block with an identifier; and

placing the identifier in a partial block list (302), said step of filling the second part including accessing the partial block list (302), with the input client (18), to determine a space in memory (24, 34) to which the image data of the second image is to be transmitted.

9. The method of claim 7 or 8 wherein the first set of blocks (306) includes a third block, and the third block remains unfilled subsequent to the second part being filled, further comprising:

designating the third block with an identifier; and

accessing the third block identifier with the input client (18) for filling the third block with image data from the second image.

10. The method of one of claims 7 to 9 wherein the input client (18) possesses a processing capability, and the first set of blocks (306) includes a predetermined number of blocks, further comprising:
adjusting the predetermined number of blocks (306) as a function of the processing capability of the input client (18).

11. The method of one of claims 7 to 10 wherein said step of filling includes filling the first block (306) and the first part of the second block (306) with image data generated from a scan service, the scan service including a scanning apparatus (18) for converting information, disposed on a document, to the image data.

Patentansprüche

1. Verfahren zur Verwaltung der Speicherzuweisung in einem Drucksystem (10) mit Steuereinheit (44) und Speicher (24, 34), wobei die Steuereinheit (44) über einen Betriebsmittelverwalter (300) zum Verwalten des Speichers (24, 34) verfügt, zumindest ein Teil des Speichers (24, 34) in eine Mehrzahl von Blöcken (306) partitioniert ist und das Drucksystem (10) Eingabe-Klienten (18) unterstützt, wobei jeder Eingabe-Klient versucht, ein Bild oder mehrere Bilder in Form von Bilddaten im Speicher (24, 34) zu speichern, wobei das Verfahren **gekennzeichnet** ist **durch:**

   die Bereitstellung eines Identifiers für jeden Block (306), der den Ort des Blocks (306) im Speicher (24, 34) angibt,
   das Platzieren eines ersten Identifier-Satzes, der einem ersten Satz von Blöcken (306) entspricht, als Reaktion auf eine Anforderung von einem ersten Eingabe-Klienten (18) in eine Datenbank (304),
   den Zugriff auf den ersten Identifier-Satz mit dem ersten Eingabe-Klienten (18),
   das Füllen eines oder mehrerer Blöcke (306) aus dem ersten Satz Blöcke (306) mit Bilddaten vom Eingabe-Klienten (18), indem auf den ersten Identifier-Satz Bezug genommen wird,
   das Übertragen eines Unterbrechungssignals mit dem ausgewählten Eingabe-Klienten (18) an die Steuereinheit (44) immer dann, wenn einer der Blöcke (306) aus dem ersten Satz (306) gefüllt ist, und
   das Platzieren eines zweiten Identifier-Satzes, der einem zweiten Satz von Blöcken (306) entspricht, in die Datenbank (304), wenn ein vorgegebener Block aus dem ersten Satz von Blöcken (306) gefüllt worden ist, wobei der Eingabe-Klient (18) nach dem Füllen eines letzten Blocks (306) aus dem ersten Satz von Blöcken (306) in die Lage versetzt wird, auf den zweiten Satz von Blöcken zuzugreifen, indem er auf den zweiten Identifier-Satz Bezug nimmt, und den zweiten Satz von Blöcken (306) zu füllen, wodurch die Speicherzuweisung mit einem Minimum an Kommunikation zwischen einzelnen Komponenten des Drucksystems (10) erfolgt.

2. Verfahren nach Anspruch 1, wobei der Schritt des Füllens das Füllen des ersten Satzes von Blöcken (306) mit von einem Scan-Service erzeugten Bilddaten umfasst, wobei der Scan-Service über eine Scan-Vorrichtung (18) zum Umwandeln der Informationen auf einem Dokument in die Bilddaten verfügt.

3. Verfahren nach Anspruch 1 oder 2, wobei der erste Satz von Blöcken (306) einen letzten Block aufweist, und
   der Schritt des Platzierens des zweiten Identifier-Satzes das Platzieren des zweiten Satzes von Identifiers in die Datenbank (304) umfasst, nachdem der letzte Block mit Bilddaten gefüllt worden ist.

4. Verfahren nach einem der Ansprüche 1 bis 3, welches zudem die folgenden Schritte aufweist:
   die Ausgabe eines entsprechenden Jobs mit einem oder beiden aus dem ersten und zweiten Satz von Blöcken (306) mit einem Ausgabe-Klienten (20).

5. Verfahren nach einem der Ansprüche 1 bis 4, wobei ein gesamter gefüllter Block zum Drucken eines Teils eines entsprechenden Jobs verwendet wird und zu dem Verfahren weiterhin gehört:
   das Platzieren eines zu dem verwendeten, gefüllten Block gehörenden Identifiers in eine erste Liste (302).

6. Verfahren nach einem der Ansprüche 1 bis 5, welches zudem umfasst:
   das erneute Partitionieren des Speichers (24, 34), nachdem eine vorgegebene Anzahl von Blöcken (306) gefüllt worden ist, damit die Blockgröße entsprechend einer variierenden Bildgröße einzelner Bilder im Speicher (24, 34) verändert wird.

7. Verfahren zur Verwaltung der Speicherzuweisung in einem Drucksystem (10) mit Speicher (24, 34), wobei zumindest ein Teil des Speichers (24, 34) in eine Mehrzahl von Blöcken (306) partitioniert wird, das Drucksystem (10) Eingabe-Klienten (18) unterstützt, ein Eingabe-Klient (18) ein erstes und ein zweites Bild in Form von Bilddaten im Speicher (24, 34) speichert, wobei das Verfahren **gekennzeichnet** ist **durch:** die Bereitstellung eines ersten Satzes von Blöcken (306) mit einem ersten und einem zweiten Block für den Eingabe-Klienten (18), wobei der zweite Block einen ersten und einen zweiten Teil einschließt, das Füllen sowohl des ersten Blocks als auch des ersten Teils des zweiten Blocks mit Bilddaten des ersten Bildes, wobei ein Ende des ersten Teils des zweiten Blocks ein entsprechendes Ende des ersten Bildes darstellt, und das Füllen des zweiten Teils des zweiten Blocks mit Bilddaten des zweiten Bildes, wobei sich das erste Bild von dem zweiten unterscheidet, wodurch der Speicherplatz zum Speichern von Bilddaten maximal ausgenutzt wird.

8. Verfahren nach Anspruch 7, zu dem weiterhin gehört:

   das Kennzeichnen des zweiten Teils des zweiten Blocks mit einem Identifier und
   das Platzieren des Identifiers in eine Teilblock-

liste (302), wobei der Schritt des Füllens des zweitens Teils das Auswerten der Teilblockliste (302) mit dem Eingabe-Klienten (18) einschließt, um dadurch einen Platz im Speicher (24, 34) zu bestimmen, zu dem die Bilddaten des zweiten Bildes übertragen werden.

9. Verfahren nach Anspruch 7 oder 8, wobei der erste Satz von Blöcken (306) einen dritten Block einschließt, und der dritte Block ungefüllt bleibt, nachdem der zweite Teil gefüllt worden ist, welches weiterhin umfasst:

das Kennzeichnen des dritten Blocks mit einem Identifier und
das Zugreifen auf den Identifier des dritten Blocks mit dem Eingabe-Klienten (18) zum Füllen des dritten Blocks mit Bilddaten vom zweiten Bild.

10. Verfahren nach einem der Ansprüche 7 bis 9, wobei der Eingabe-Klient (18) ein Verarbeitungsvermögen besitzt und der erste Satz von Blöcken (306) eine vorgegebene Anzahl von Blöcken einschließt, welches weiterhin umfasst:

das Einstellen einer vorgegebenen Anzahl von Blöcken (306) als Funktion des Verarbeitungsvermögens des Eingabe-Klienten (18).

11. Verfahren nach einem der Ansprüche 7 bis 10, wobei zum Schritt des Füllens das Füllen des ersten Blocks (306) und des ersten Teils des zweiten Blocks (306) mit von einem Scan-Service erzeugten Bilddaten gehört, wobei der Scan-Service eine Scan-Vorrichtung (18) zum Umwandeln von Informationen auf einem Dokument in die Bilddaten aufweist.

Revendications

1. Procédé de gestion d'une allocation de mémoire dans un système d'impression (10) comprenant un contrôleur (44) et une mémoire (24, 34), le contrôleur (44) comportant un gestionnaire de ressources (300) destiné à gérer l'utilisation de la mémoire (24, 34), au moins une partie de la mémoire (24, 34) étant segmentée en une pluralité de blocs (306), le système d'impression (10) acceptant des clients en entrée (18), chaque client en entrée (18) cherchant à mémoriser une ou plusieurs images, sous forme de données d'image, dans la mémoire (24, 34), caractérisé par les étapes suivantes :

fournir à chaque bloc (306) un identificateur indiquant un emplacement du bloc (306) dans la mémoire (24, 34),
en réponse à une demande provenant de l'un des clients en entrée (18), placer un premier ensemble d'identificateurs, correspondant à un premier ensemble de blocs (306), dans une base de données (304),
accéder au premier ensemble d'identificateurs avec le client en entrée (18),
remplir un ou plusieurs blocs (306) du premier ensemble de blocs (306), avec des données d'image provenant du client en entrée (18), en se référant au premier ensemble d'identificateurs,
transmettre un signal d'interruption avec le client en entrée (18) au contrôleur (44) à chaque fois qu'un bloc (306) du premier ensemble de blocs (306) est rempli, et
en réponse à la réception du signal d'interruption au niveau du contrôleur (44), placer un second ensemble d'identificateurs, correspondant à un second ensemble de blocs (306), dans la base de données (304), dans le cas où un bloc prédéterminé (306) du premier ensemble de blocs (306) a été rempli, dans lequel le client en entrée (18) après le remplissage d'un dernier bloc (306) du premier ensemble de blocs (306), est autorisé à accéder au second ensemble de blocs (306), en se référant au second ensemble d'identificateurs, et à remplir des blocs (306) du second ensemble de blocs (306),
en accomplissant ainsi une allocation de mémoire avec une quantité minimum de communications à l'intérieur du système d'impression (10).

2. Procédé selon la revendication 1, dans lequel : ladite étape de remplissage comprend le remplissage du premier ensemble de blocs (306) avec des données d'image générées à partir d'un service de numérisation, le service de numérisation comprenant un dispositif de numérisation (18) destiné à convertir des informations, disposées sur un document, en données d'image.

3. Procédé selon la revendication 1 ou 2, dans lequel :

le premier ensemble de blocs (306) comprend un dernier bloc, et
ladite étape consistant à placer le second ensemble d'identificateurs comprend le fait de placer le second ensemble d'identificateurs dans la base de données (304) après que le dernier bloc a été rempli avec des données d'image.

4. Procédé selon l'une des revendications 1 à 3, comprenant en outre :

la fourniture en sortie d'une tâche correspondante, comprenant un ou plusieurs des premier et

second ensembles de blocs (306), avec un client en sortie (20).

5. Procédé selon l'une des revendications 1 à 4, dans lequel la totalité d'un bloc rempli est utilisée pour imprimer une partie d'une tâche correspondante, comprenant en outre :

le fait de placer un identificateur, associé au bloc rempli utilisé, dans une première liste (302).

6. Procédé selon l'une des revendications 1 à 5, comprenant en outre :

un nouveau cloisonnement de la mémoire (24, 34), après qu'un nombre prédéterminé de blocs (306) ont été remplis, en vue de modifier la taille des blocs conformément à une taille d'image variable des images individuelles qui sont mémorisées dans une mémoire (24, 34).

7. Procédé de gestion d'une allocation de mémoire dans un système d'impression (10) comprenant une mémoire (24, 34), au moins une partie de la mémoire (24, 34) étant cloisonnée en une pluralité de blocs (306), le système d'impression (10) acceptant des clients en entrée (18), un client en entrée (18) mémorisant une première image et une seconde image, sous forme de données d'image, dans la mémoire (24, 34), caractérisé par les étapes suivantes :

fournir au client en entrée (18) un premier ensemble de blocs (306) comprenant un premier bloc et un second bloc, le second bloc comprenant une première partie et une seconde partie,
remplir à la fois le premier bloc et la première partie du second bloc avec des données d'image de la première image, une extrémité de la première partie du second bloc représentant une extrémité correspondante de la première image, et
remplir la seconde partie du second bloc avec des données d'images de la seconde image, dans laquelle la première image est différente de la seconde image en maximisant ainsi l'utilisation de l'espace mémoire pour mémoriser les données d'image.

8. Procédé selon la revendication 7, comprenant en outre :

la désignation de la seconde partie du second bloc par un identificateur, et
le fait de placer l'identificateur dans une liste de blocs partiels (302), ladite étape de remplissage de la seconde partie comprenant un accès à la liste des blocs partiels (302), avec le client en entrée (18), afin de déterminer un espace dans la mémoire (24, 34) vers lequel les données d'image de la seconde image doivent être transmises.

9. Procédé selon la revendication 7 ou 8, dans lequel le premier ensemble de blocs (306) comprend un troisième bloc, et le troisième bloc reste non rempli après que la seconde partie est remplie, comprenant en outre :

la désignation du troisième bloc avec un identificateur, et
un accès à l'identificateur du troisième bloc avec le client en entrée (18) en vue de remplir le troisième bloc avec des données d'image provenant de la seconde image.

10. Procédé selon l'une des revendications 7 à 9, dans lequel le client en entrée (18) possède une capacité de traitement, et le premier ensemble de blocs (306) comprend un nombre prédéterminé de blocs, comprenant en outre :

l'ajustement du nombre prédéterminé de blocs (306) en fonction de la capacité de traitement du client en entrée (18).

11. Procédé selon l'une des revendications 7 à 10, dans lequel ladite étape de remplissage comprend le remplissage du premier bloc (306) et de la première partie du second bloc (306) avec des données d'image générées à partir d'un service de numérisation, le service de numérisation comprenant un dispositif de numérisation (18) destiné à convertir des informations, disposées sur un document, en les données d'image.

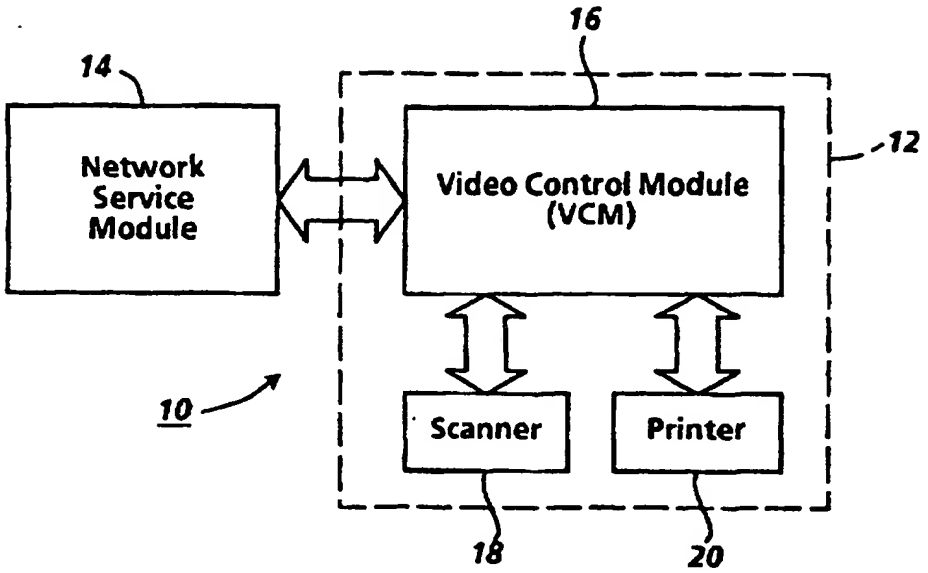
16
14
Network Service Module
Video Control Module (VCM)
12
10
Scanner
Printer
18
20

FIG. 1

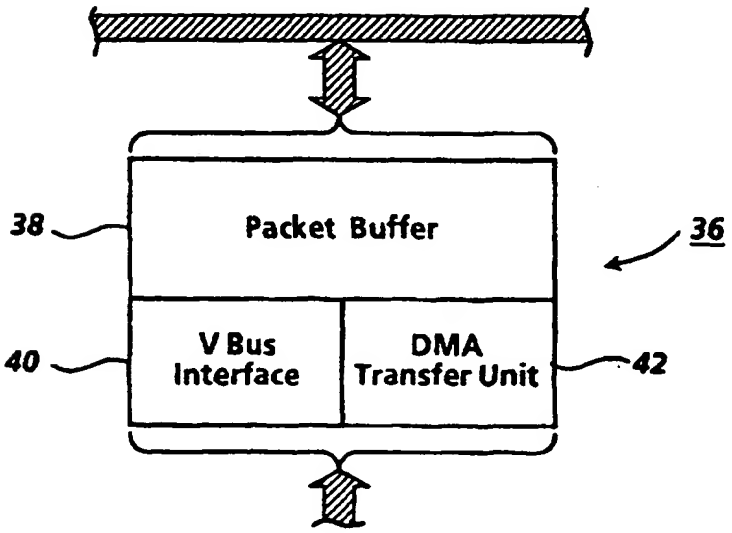
38
Packet Buffer
36
40
V Bus Interface
DMA Transfer Unit
42

FIG. 3

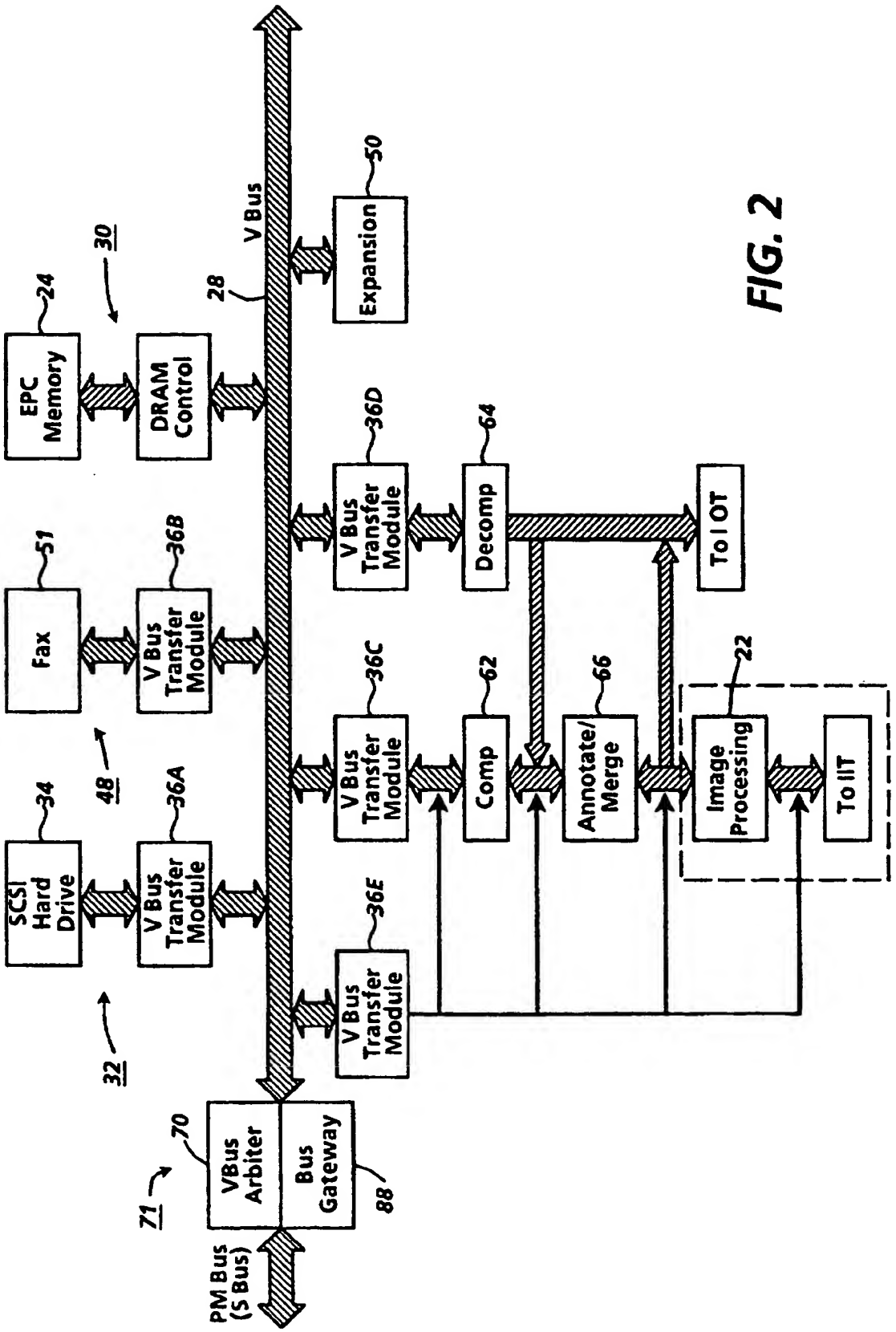
PM Bus (S Bus)
71
70
VBus Arbiter
Bus Gateway
88
32
SCSI Hard Drive
34
V Bus Transfer Module
36A
48
Fax
51
V Bus Transfer Module
36B
EPC Memory
24
DRAM Control
30
28
V Bus
Expansion
50
V Bus Transfer Module
36E
V Bus Transfer Module
36C
V Bus Transfer Module
36D
Comp
62
Decomp
64
Annotate/ Merge
66
Image Processing
22
To IIT
To IOT

FIG. 2

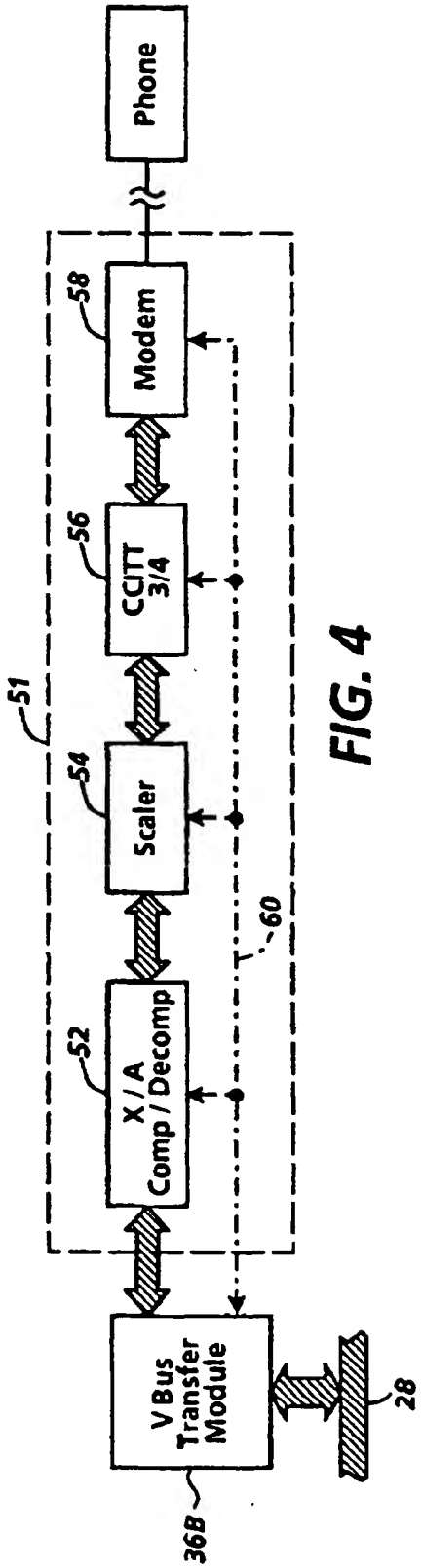
Phone
58
Modem
56
CCITT 3/4
54
Scaler
52
X/A Comp/Decomp
51
60
V Bus Transfer Module
28
36B

FIG. 4

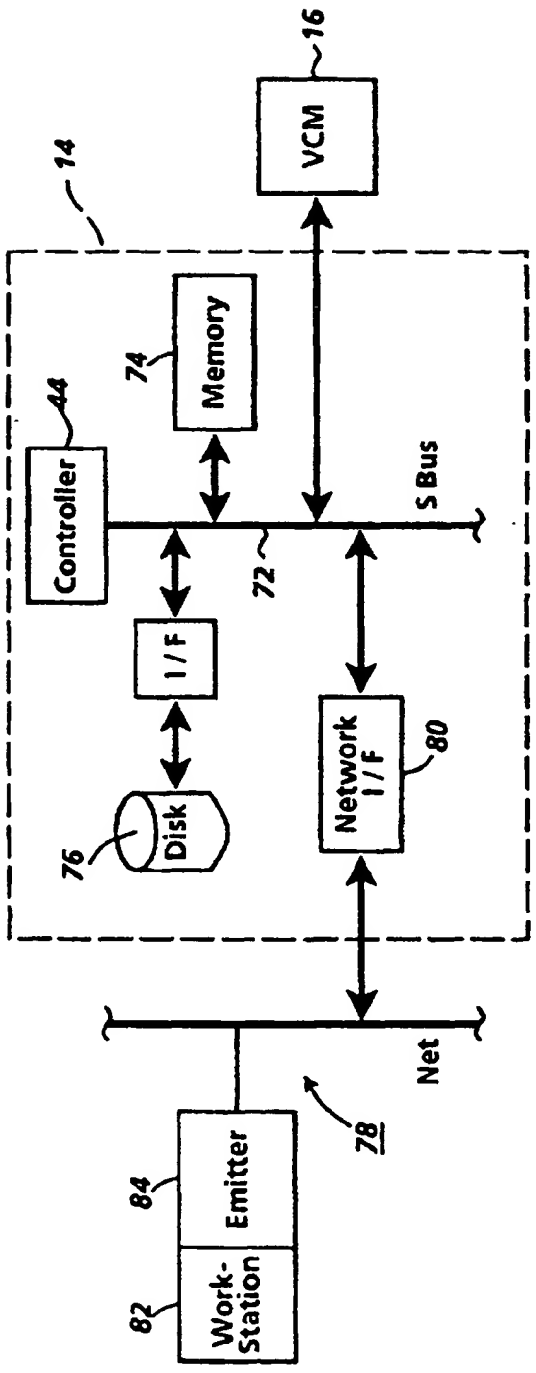
16
VCM
14
74
Memory
44
Controller
S Bus
72
I/F
Network I/F
80
76
Disk
Net
78
84
Emitter
82
Work-Station

FIG. 5

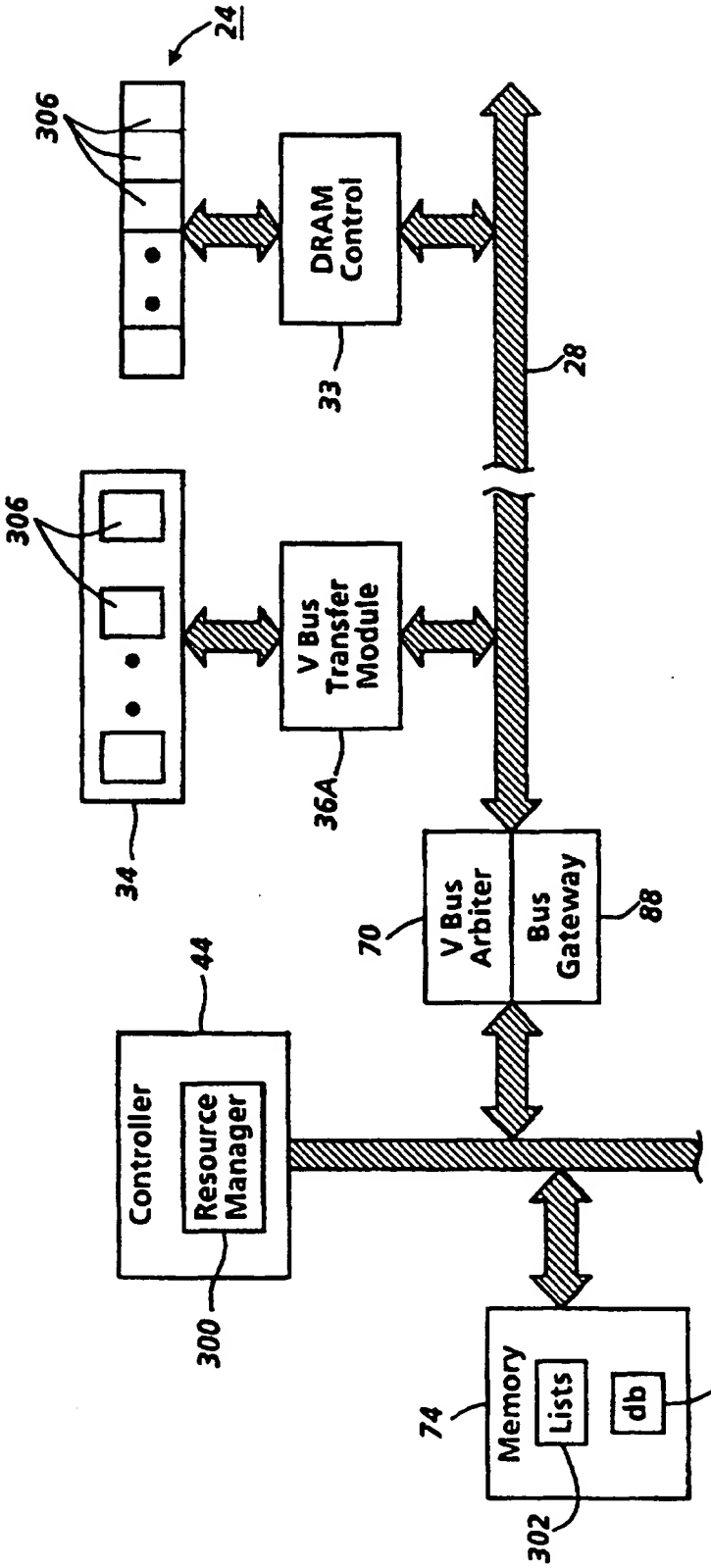
24
306
DRAM Control
33
28
306
V Bus Transfer Module
36A
34
70
V Bus Arbiter
Bus Gateway
88
44
Controller
Resource Manager
300
74
Memory
Lists
db
304
302

FIG. 6

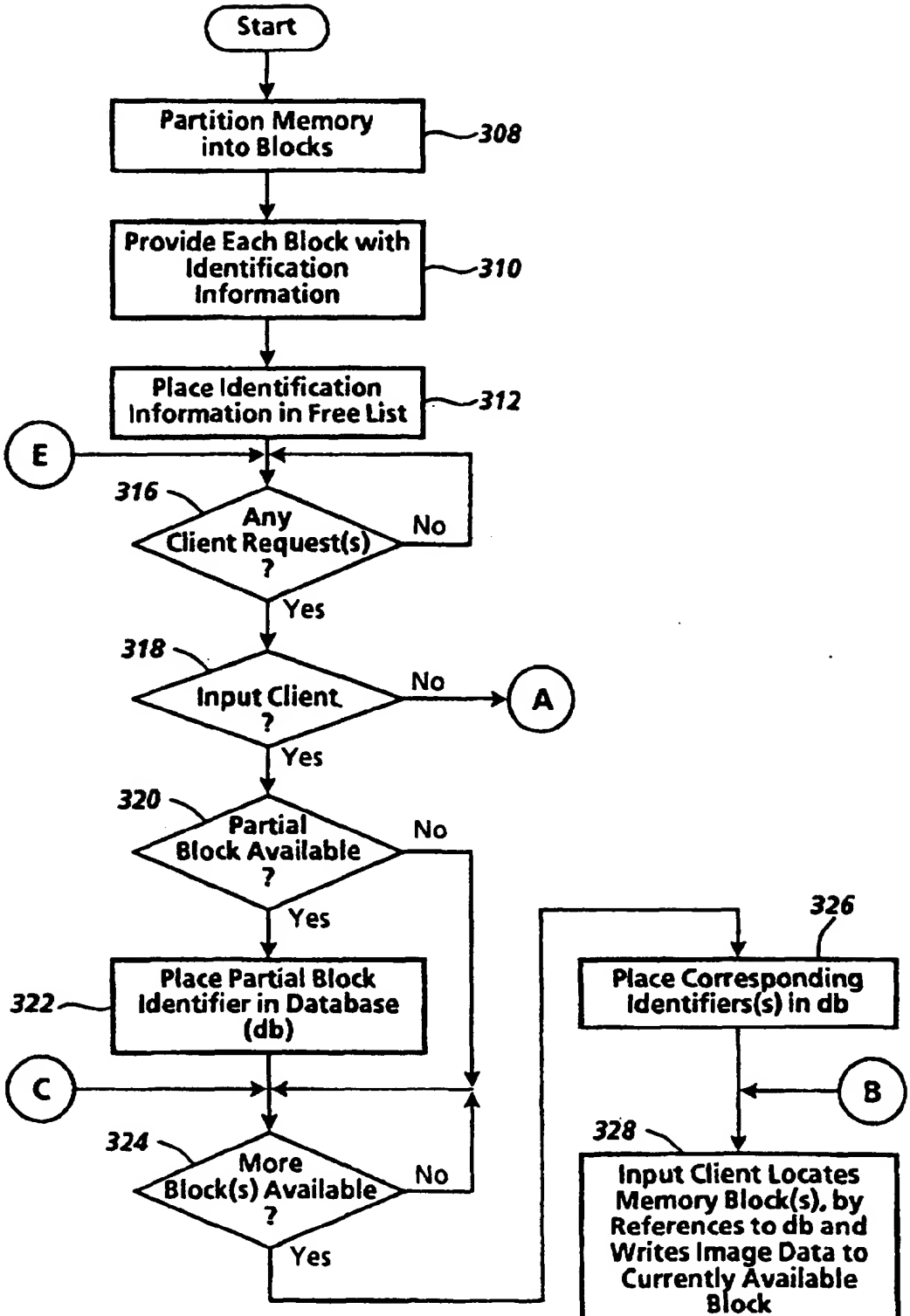
Start
Partition Memory into Blocks
308
Provide Each Block with Identification Information
310
Place Identification Information in Free List
312
E
316
Any Client Request(s) ?
No
Yes
318
Input Client ?
No
A
Yes
320
Partial Block Available ?
No
Yes
322
Place Partial Block Identifier in Database (db)
C
324
More Block(s) Available ?
No
Yes
326
Place Corresponding Identifiers(s) in db
B
328
Input Client Locates Memory Block(s), by References to db and Writes Image Data to Currently Available Block
To Fig. 8

FIG. 7

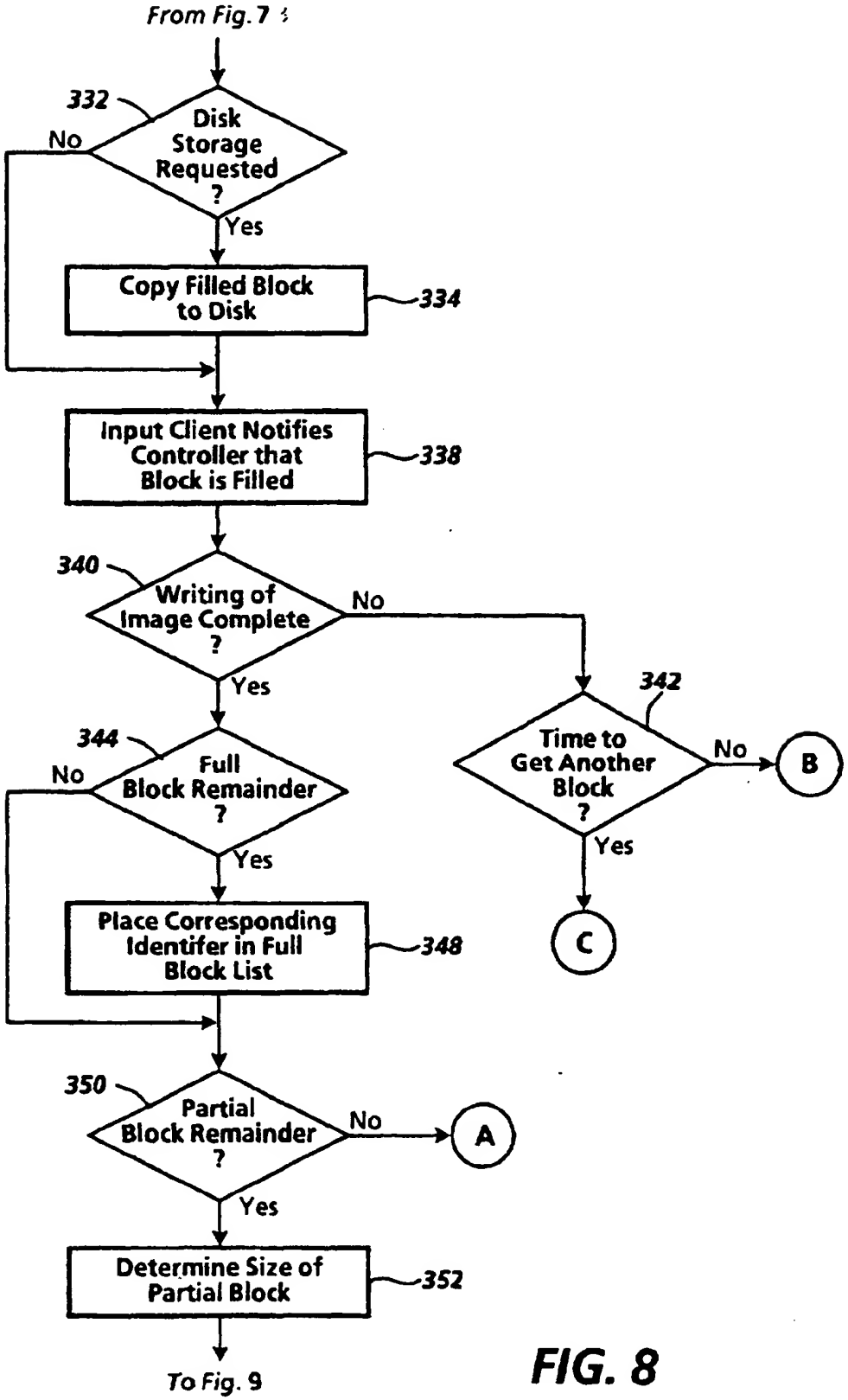
From Fig. 7
332
Disk Storage Requested ?
No
Yes
Copy Filled Block to Disk
334
Input Client Notifies Controller that Block is Filled
338
340
Writing of Image Complete ?
No
Yes
342
Time to Get Another Block ?
No
B
Yes
C
344
Full Block Remainder ?
No
Yes
Place Corresponding Identifer in Full Block List
348
350
Partial Block Remainder ?
No
A
Yes
Determine Size of Partial Block
352
To Fig. 9

FIG. 8

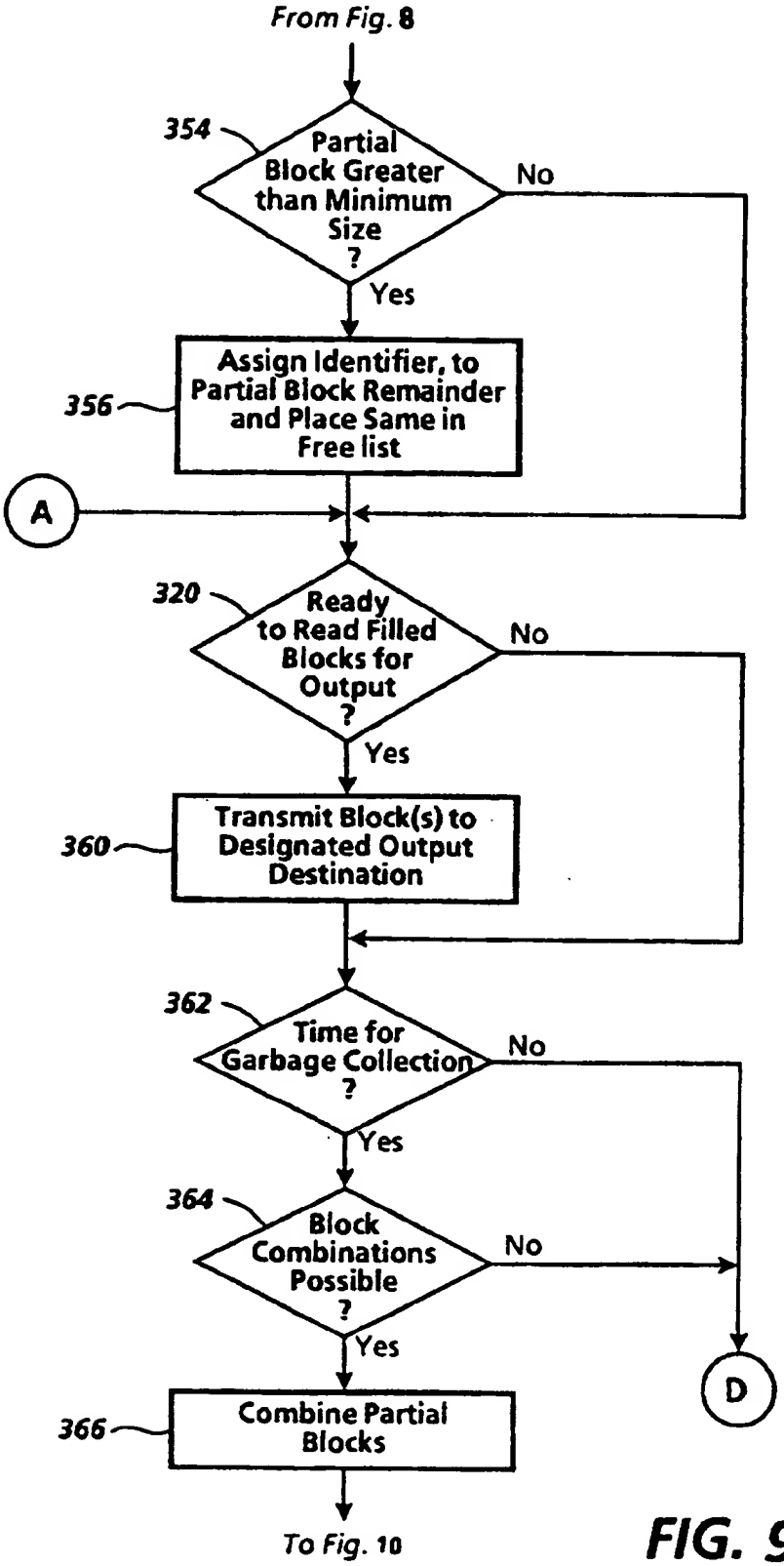
From Fig. 8
354
Partial Block Greater than Minimum Size ?
No
Yes
356
Assign Identifier, to Partial Block Remainder and Place Same in Free list
A
320
Ready to Read Filled Blocks for Output ?
No
Yes
360
Transmit Block(s) to Designated Output Destination
362
Time for Garbage Collection ?
No
Yes
364
Block Combinations Possible ?
No
Yes
D
366
Combine Partial Blocks
To Fig. 10

FIG. 9

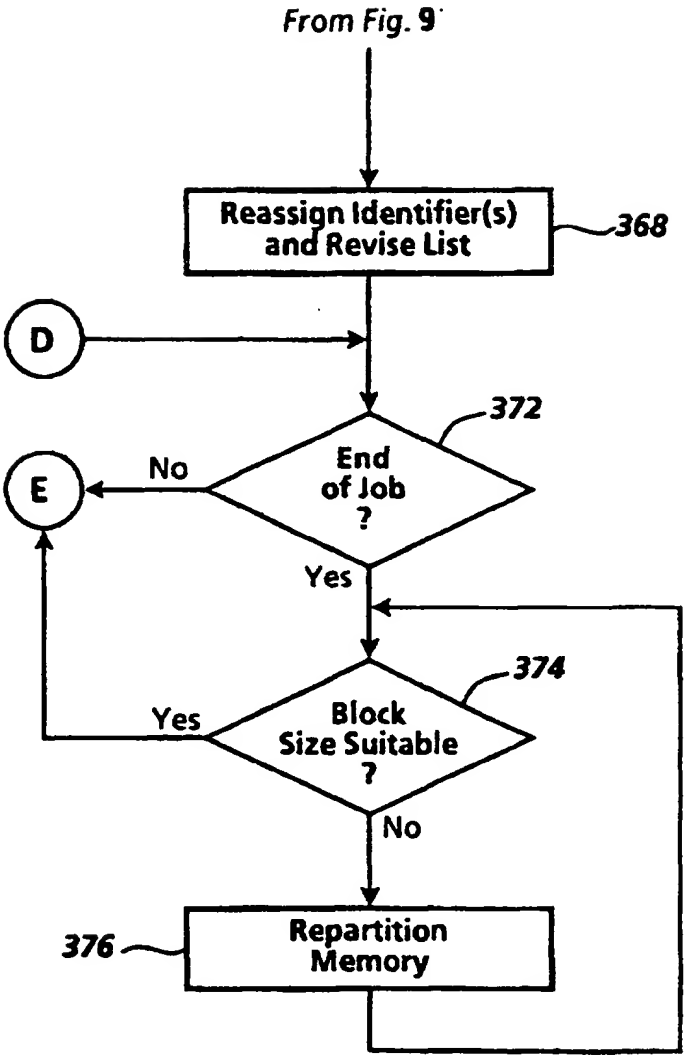
From Fig. 9
Reassign Identifier(s) and Revise List
368
D
372
End of Job ?
No
E
Yes
374
Block Size Suitable ?
Yes
No
376
Repartition Memory

FIG. 10

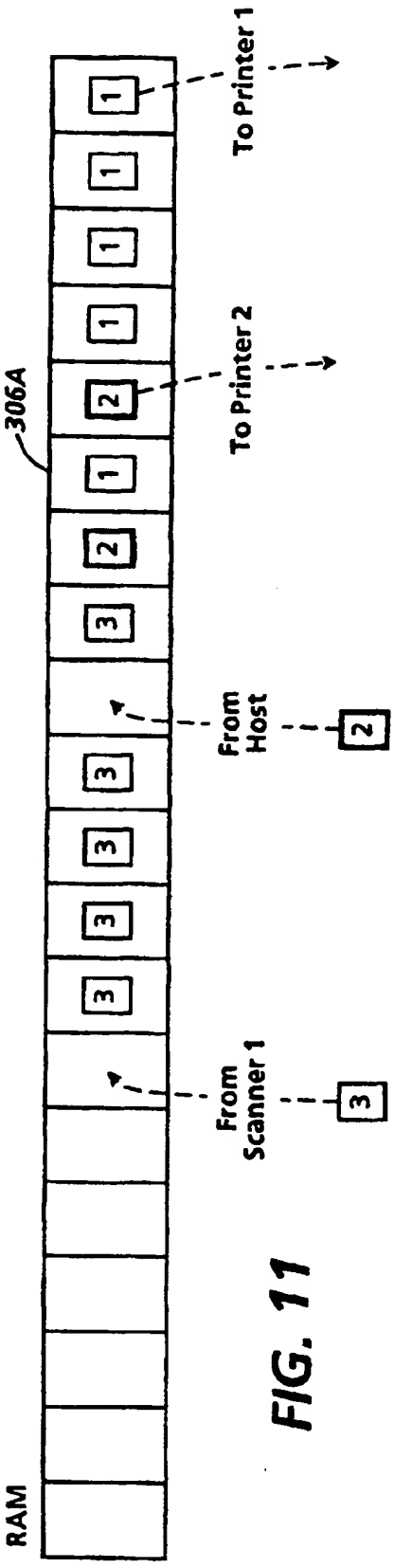
RAM
306A
From Scanner 1
From Host
To Printer 2
To Printer 1

FIG. 11

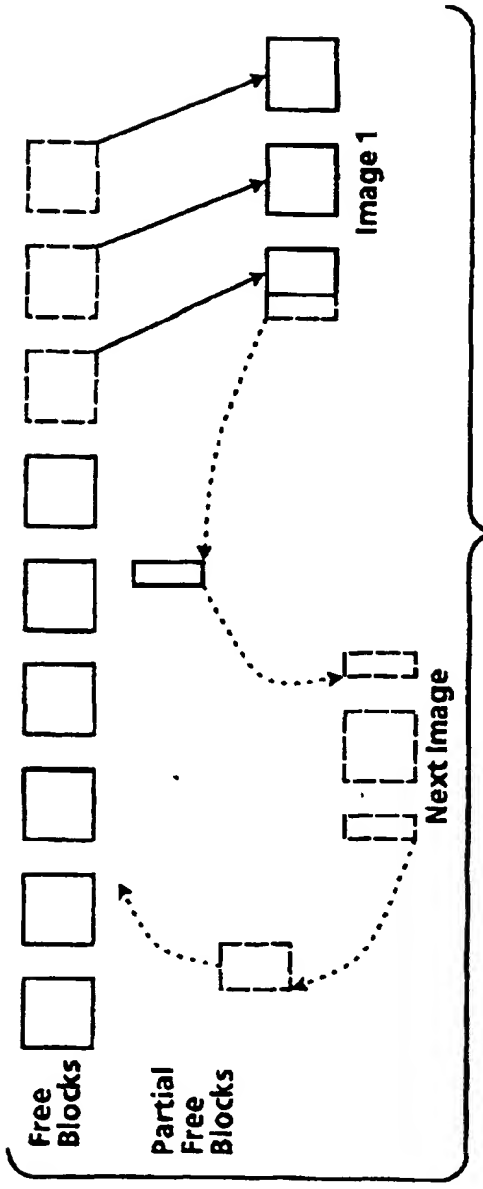
Free Blocks
Partial Free Blocks
Next Image
Image 1

FIG. 12

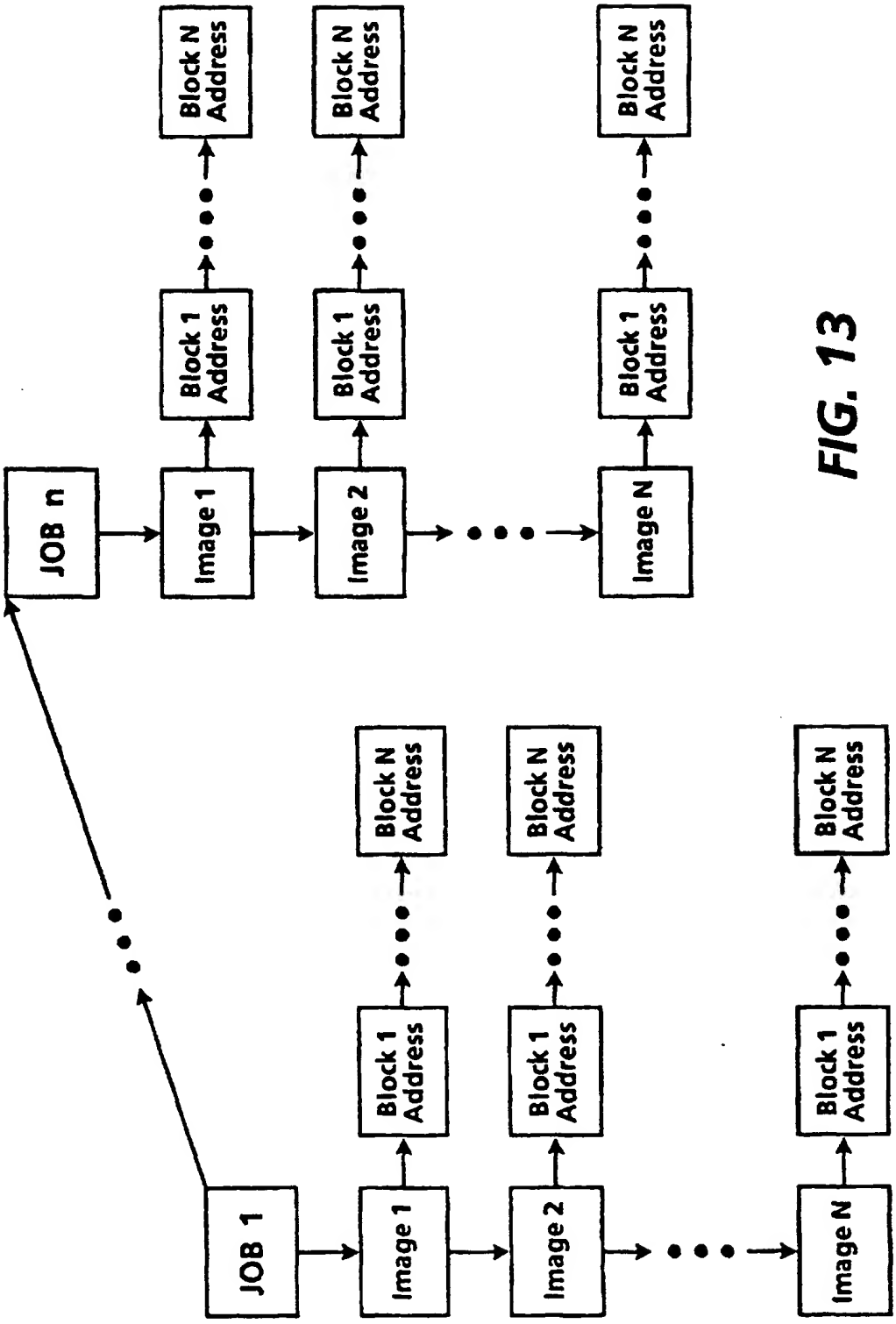
JOB n
Image 1
Image 2
Image N
Block 1 Address
Block N Address
JOB 1
Image 1
Image 2
Image N
Block 1 Address
Block N Address

FIG. 13

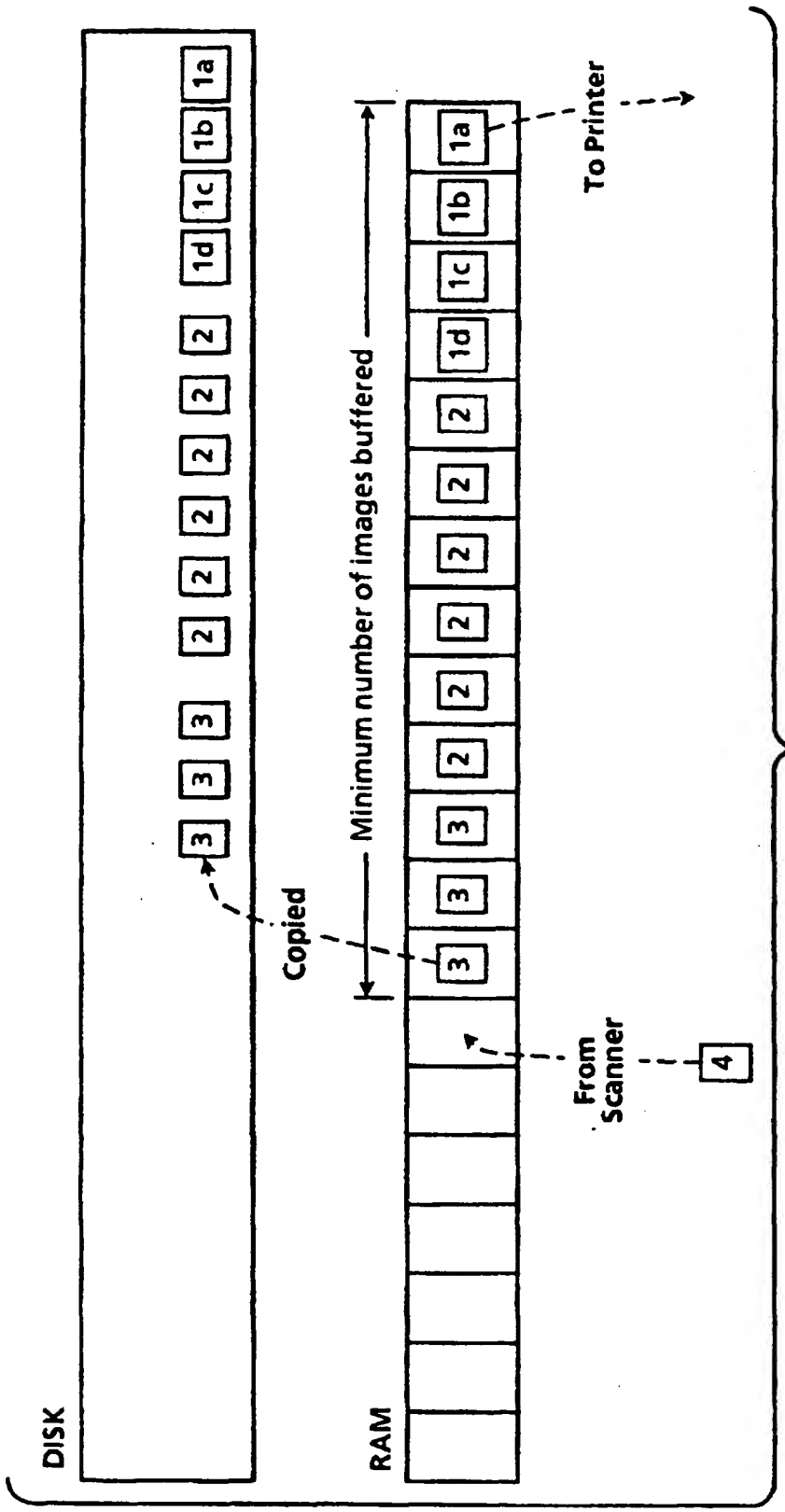
DISK
RAM
Copied
Minimum number of images buffered
From Scanner
To Printer
1a
1b
1c
1d
2
3
4

FIG. 14

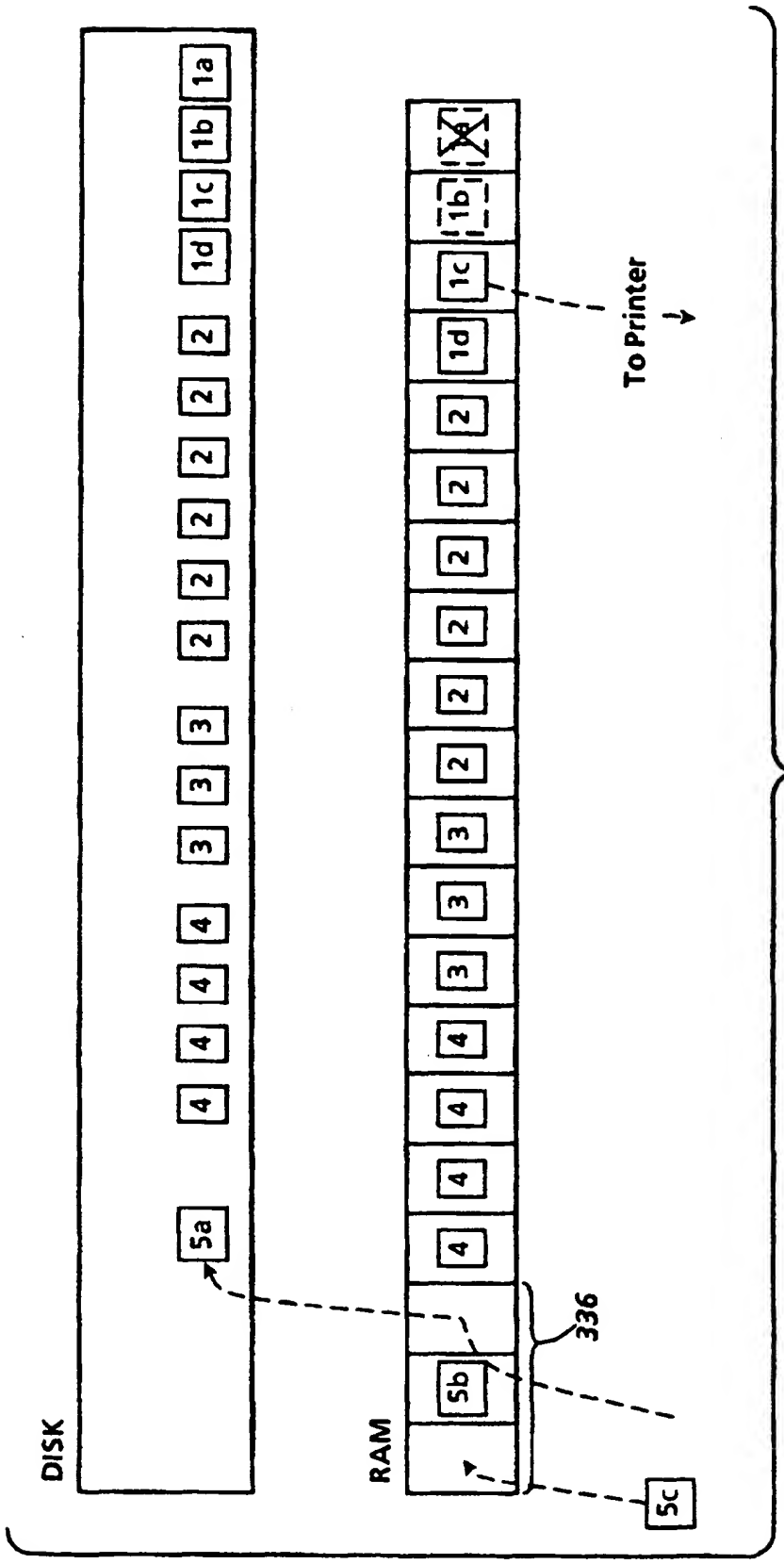

FIG. 15

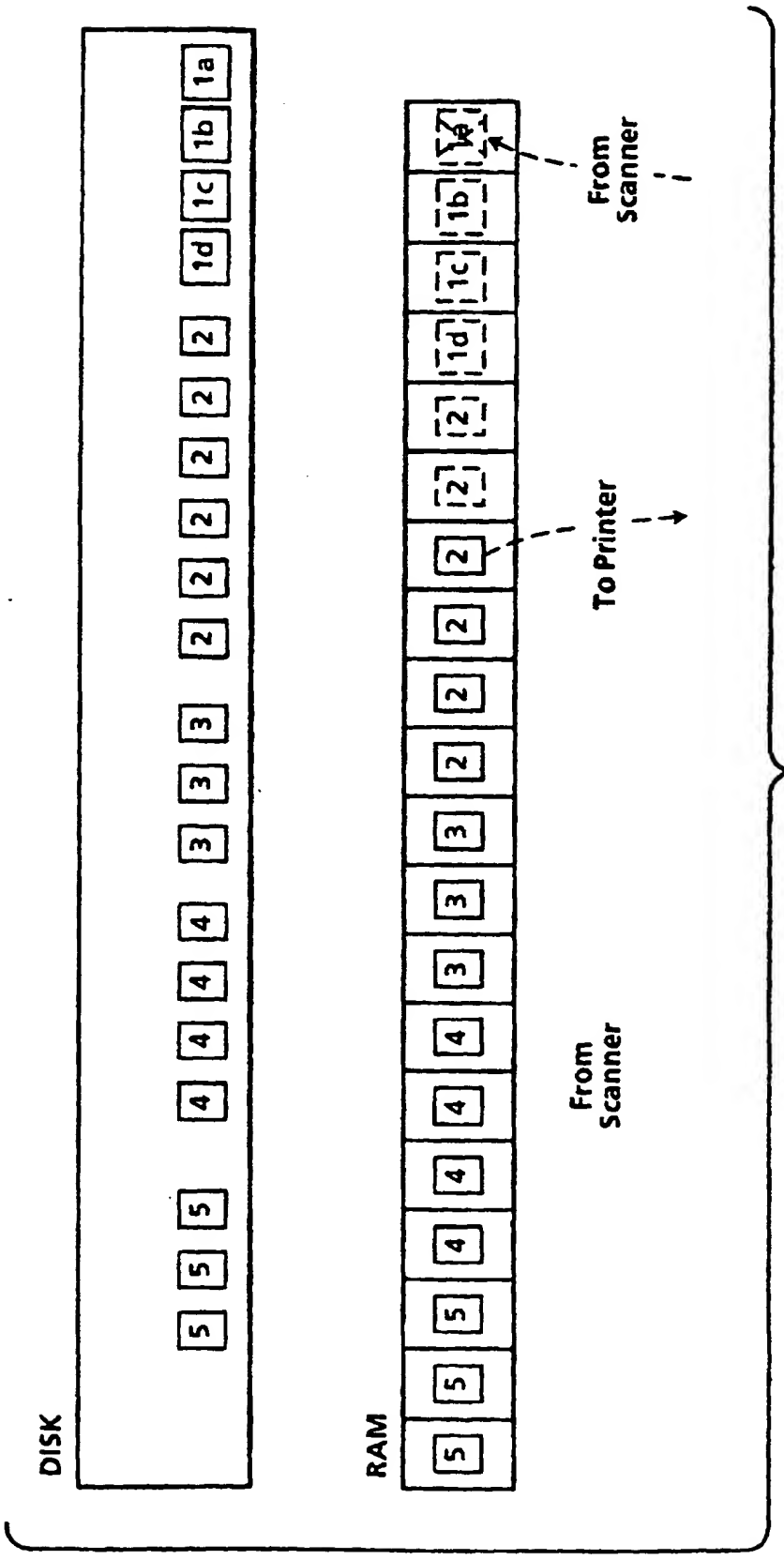

FIG. 16

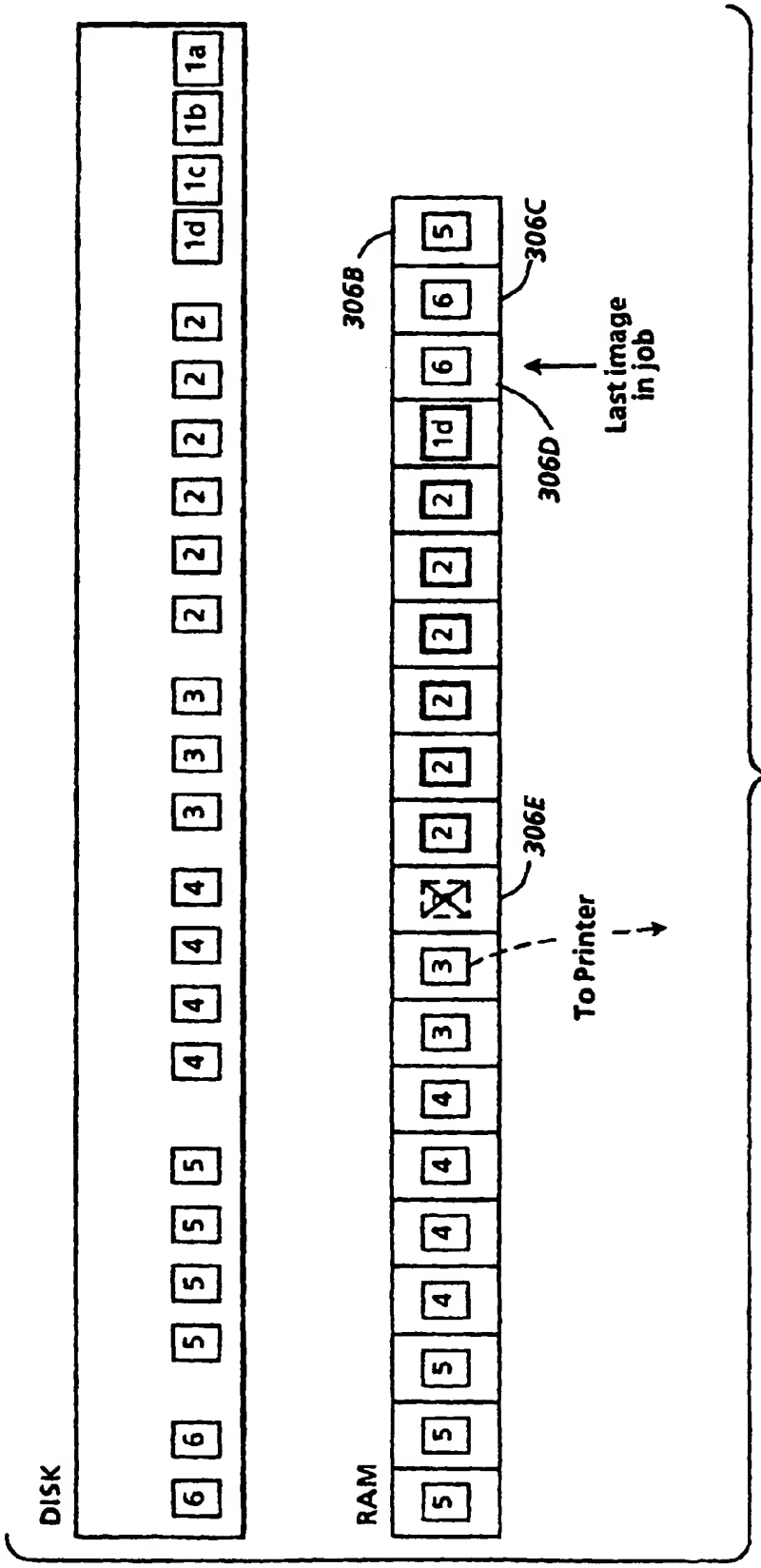
DISK
6 6 5 5 5 5 4 4 4 4 3 3 3 2 2 2 2 2 2 2 1d 1c 1b 1a
RAM
5 5 5 4 4 4 4 4 3 3 2 2 2 2 2 2 2 1d 6 6 5
306B
306C
306D
306E
Last image in job
To Printer

FIG. 17